INPUT
8
L. 2800
CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER
PAPER FEED
COLOR CHANGE
PEN CHANGE
ISTITUTO GEOGRAFICO DE AGOSTINI

CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER

Pubblicazione a fascicoli settimanali
edita dall'Istituto Geografico De Agostini

volume I - fascicolo 8

Direttori: Achille Boroli - Adolfo Boroli

Direzione editoriale: Mario Nilo; *settore fascicoli:* Jason Vella

Redazione dell'edizione italiana a cura della:
Logical Studio Communication

Traduzione dall'inglese a cura di: Daniel Quinn

Coordinamento grafico: Otello Geddo

Coordinamento fotografico a cura del Centro Iconografico dell'Istituto Geografico De Agostini

Direzione:
Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5

Redazione:
Milano (20149), via Mosè Bianchi 6 - tel. (02) 4694451

Amministrazione, abbonamenti e servizio arretrati:
Istituto Geografico De Agostini - Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5.

Copertine e risguardi per i volumi dell'opera saranno messi in vendita a L. 6000 la copia (L. 7500 all'estero).

Le copie arretrate saranno disponibili per un anno dal completamento dell'opera e potranno essere prenotate nelle edicole o direttamente presso l'Editore. Per i fascicoli arretrati, trascorse 12 settimane dalla loro pubblicazione, è applicato un sovrapprezzo di L. 400 sul prezzo di copertina in vigore al momento dell'evasione dell'ordine. Spedizione contro rimessa di pagamento anticipato; non vengono effettuate spedizioni contrassegno.

L'Editore si riserva la facoltà di modificare il prezzo nel corso della pubblicazione, se costretto da mutate condizioni di mercato.

Registrato presso il Tribunale di Novara n. 11 in data 19-5-1984.

Direttore responsabile: Emilio Bucciotti

Spedizione in abbonamento postale Gruppo II/70 (Autorizzazione della Direzione provinciale delle PP.TT. di Novara).

Distribuzione A. & G. Marco - Milano, via Fortezza 27 - tel. (02) 2526.

Pubblicazione a fascicoli settimanali. Esce il martedì.

Stampato in Italia - I.G.D.A. Officine Grafiche, Novara - 118412.

Referenze dei disegni e delle fotografie:
Copertina: Dave King. Pagg. 225, 226, 228 Nick Farmer. Pagg. 230, 232, 233, 234, 235 Tudor Art Studios. Pagg. 236, 237, 238 Chris Lyon. Pagg. 240, 242, 244 Dave King/Ian Craig/Roy Flukes. Pagg. 243, 245, 246 Ian Stephen. Pag. 248 Associated Press/Graham Bingham. Pagg. 250, 252, 254 Jd Audio Visual. Pagg. 255 Ray Duns. Apparecchiature messe a disposizione da Lasky's Tottenham Court Road, London W1.

INPUT È STUDIATA APPOSITAMENTE PER:

Lo SPECTRUM della Sinclair (versioni 16K e 48K), il COMMODORE 64, l'ELECTRON ed il BBC della ACORN, il DRAGON 32.

Comunque, molti dei programmi e dei testi sono adatti anche per: lo ZX81 della SINCLAIR, il COMMODORE VIC 20 ed il TANDY COLOUR COMPUTER con 32K ed il BASIC esteso.

I seguenti simboli identificano i programmi o le spiegazioni adatte a ciascun computer:

 SPECTRUM

 COMMODORE 64

 ELECTRON e BBC

 DRAGON 32

 ZX81

 VIC 20

 TANDY TRS80 COLOUR COMPUTER

STAMPANTI: QUALE SCEGLIERE?

- QUANDO SERVE UNA STAMPANTE?
- LA SCELTA DEL TIPO GIUSTO
- L'ACQUISTO DELLA CARTA
- IL COLLEGAMENTO DELLA STAMPANTE AL COMPUTER

Se si desidera una copia duratura di un programma o di un disegno sullo schermo allora serve una stampante. Ma quale scegliere?

L'acquisto di una stampante consente di utilizzare appieno il proprio computer e di incrementare le proprie capacità di programmatori.

Poiché si tratta di un investimento non indifferente, occorre essere ben informati sulle caratteristiche dei vari modelli, prima di procedere all'acquisto.

Di norma, un computer emette informazioni per l'utente tramite lo schermo di un'apparecchio TV o di un monitor. Una stampante costituisce un mezzo alternativo per visualizzare tali informazioni. Sebbene simili a prima vista, i due dispositivi non hanno, in realtà, lo stesso scopo: la peculiarità di una stampante è di offrire una copia su carta, una registrazione durevole di tutto ciò che viene riprodotto, solo temporaneamente, sullo schermo.

Le occasioni per usare una stampante non mancano certo: immagini di grafica (magari di *computer art*), degne di essere incorniciate, oppure lunghi listati di programma che, in forma stampata, sono più comodi da rivedere e correggere, anche quando si è lontani dal proprio computer o, ancora, grafici e tabulati da allegare a relazioni, anch'esse prodotte su stampante.

Tutti gli home computer hanno una tastiera più o meno simile a quella di una macchina da scrivere, utile all'immissione di valori e di parole, ma le operazioni possibili su un testo possono spingersi oltre la mera e semplice scrittura, per sconfinare nel mondo del word processing, in cui il testo può essere elaborato in più modi. Con il programma adatto, ogni micro

Nelle stampanti a matrice di punti, una singola colonna di aghi smussati colpisce un nastro inchiostrato, producendo la serie di punti necessari a dare forma al carattere in stampa

può offrire prestazioni a dir poco eccezionali, ma una lettera o un opuscolo, per quanto ben composti, valgono ben poco se non si ha modo di stamparli.

Inoltre, possedere una stampante facilita anche il lavoro di programmazione. Ottenere una copia permanente del lavoro fatto su un programma risulta già di per sé un fatto prezioso ma, inoltre, è più facile localizzare errori su carta piuttosto che scorrendo il listato sullo schermo. Indubbiamente, in questo aspetto gioca molto la maggiore familiarità che ancora abbiamo con informazioni stampate rispetto a immagini elettroniche.

Questa lunga enunciazione di elogi può far sembrare indispensabile la stampante per chiunque usi un computer, ma va anche ricordato che tali pregi non sono uniti in ogni *stampante* e quelle più versatili hanno costi proibitivi. Per esempio, non tutte le stampanti sono capaci di riprodurre con precisione la grafica ottenuta sullo schermo e solo pochissime stampano in più di un colore. Prima di tutto, dunque, occorre accertare le proprie necessità e scegliere soltanto quando si è certi di ciò che si vuole. Per offrire una guida di massima all'acquisto, esploriamo le caratteristiche dei vari sistemi in commercio.

TIPI DI STAMPANTE

I tipi di stampante adatti agli home computer sono quattro: a matrice di punti, a margherita, termica e a penna. Il tipo più diffuso è quello a matrice di punti, perché risulta, in genere, anche il più economico. In quasi tutte le moderne stampanti, il carrello resta fermo, mentre una testina di stampa scorre orizzontalmente avanti e indietro lungo la larghezza del foglio. La testina contiene una fitta serie di aghi, azionati da minuscoli solenoidi. Quando viene attivato un solenoide, l'ago corrispondente viene spinto con forza tra la testina e la carta, producendo un punto. La forma dei caratteri si ottiene grazie a un'opportuna combinazione di aghi e all'avanzamento della testina di stampa.

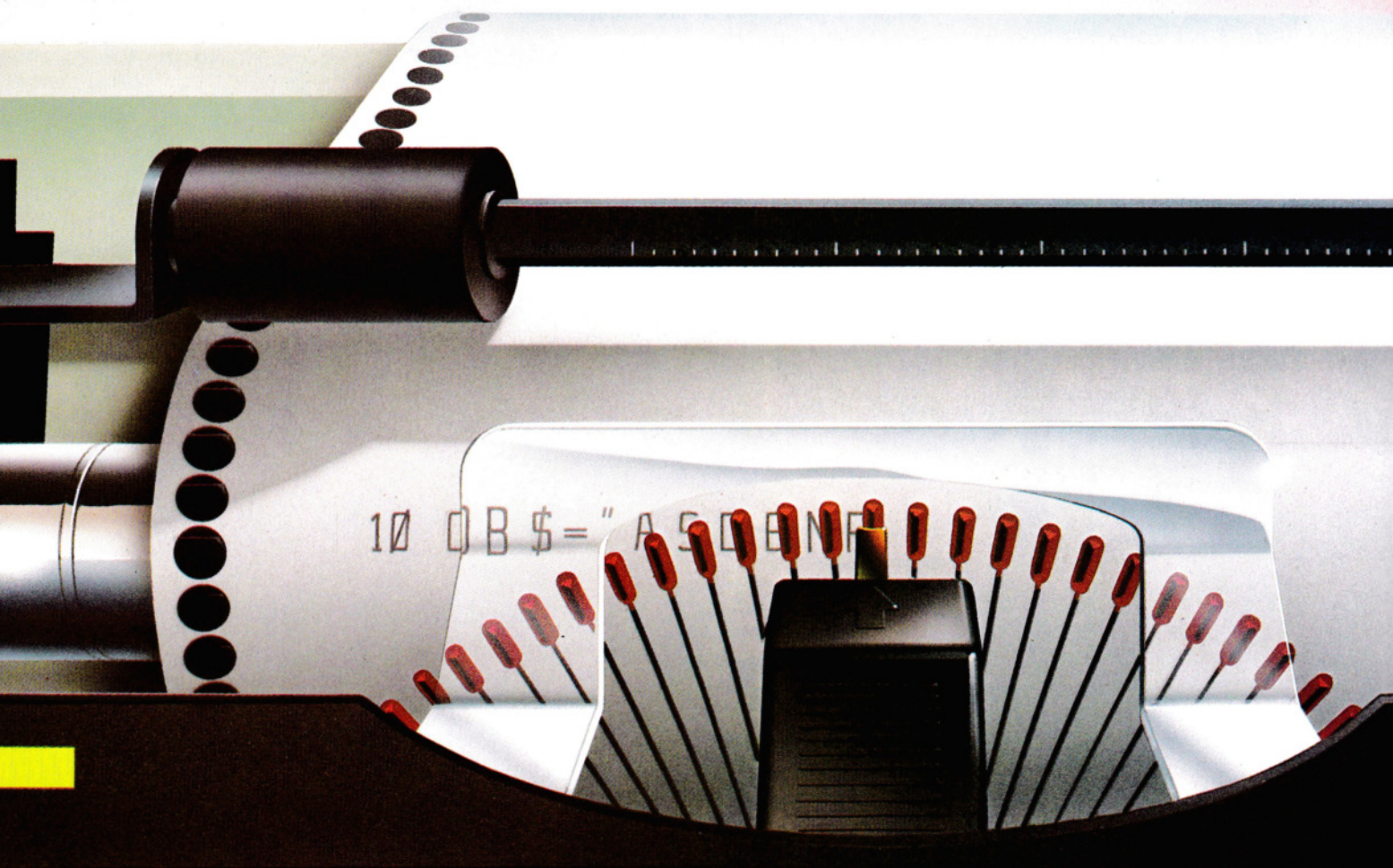
Chi possiede un Commodore può scegliere tra due stampanti a matrici di punti, diverse nel prezzo e nella resa. Il loro vantaggio è di non richiedere interfacciamento (vedere a pagina 229) e di essere capaci di stampare i caratteri grafici ROM del Commodore, al contrario di altre stampanti che creerebbero problemi, ad esempio per i listati.

Le stampanti di questo genere hanno un'alta velocità di stampa, da 8Ø caratteri al secondo (cps) fino a più di 4ØØ. Uno svantaggio di alcune di queste stampanti è che i caratteri non possiedono tratti discendenti (ossia le gambette delle lettere come ha la 'q' o la 'g'): l'intero carattere si appoggia sulla linea di stampa, senza che la gambetta passi di sotto. I tratti discendenti rendono la stampa più gradevole, ma anche più facilmente comprensibile (ad esempio, non si confondono le 's' con le 'g').

I caratteri delle prime stampanti a matrici di punti erano prodotti da una testina di stampa di sette aghi in una matrice, o modulo, di 5 colonne per 7 linee di punti. Le più recenti impiegano nove aghi per produrre una matrice di 7×9 o 9×9.

Anche con la migliore stampante di questo tipo, i caratteri risultano distintamente composti di punti, benché perfettamente leggibili. Ecco perché queste stampanti sono per lo più adatte a lavori non professionali o per riprodurre i disegni creati sullo schermo. Comunque, con l'andar del tempo la qualità di questo genere di stampanti va migliorando.

Un metodo, noto come stampa "multipass", consiste nel far ripassare la testina di stampa due volte su ogni linea del testo, prima in un senso poi tornando indie-

tro. Siccome al secondo passaggio c'è sempre una piccola sfasatura nell'allineamento, i tipici spazi vuoti tra i punti dei caratteri si riempiono, producendo caratteri più 'neri' (ma con un tempo di stampa raddoppiato!)

Un altro metodo (che non implica altrettante perdite di tempo) adotta testine con colonne di aghi multiple: i caratteri si formano in più fasi, prima la colonna di destra, poi quella a sinistra e così via.

Benché questi metodi offrano un miglioramento notevole, la qualità delle stampanti a matrici non riesce mai a superare certi livelli: quando occorre un'alta qualità (come per corrispondenze formali), la scelta migliore è una stampante a corpo solido, in cui ad ogni carattere corrisponde un martelletto sul quale è riprodotta, a rilievo, la lettera da stampare, proprio come accade nelle normali macchine da scrivere.

STAMPANTI A CARATTERE PIENO

Le prime stampanti a carattere pieno per microcompuer sono derivate da un adattamento delle macchine da scrivere elettriche a testina rotante. La qualità di stampa era eccellente, ma la massa della testina (concepita per velocità di battuta non superiore a 15 caratteri al secondo) rallentava notevolmente il processo di stampa e influiva sull'affidabilità del sistema. Nacque così la cosiddetta 'margherita': un dispositivo di stampa intercambiabile, la cui forma richiama, appunto, quella del comune fiore. In cima a ogni *petalo* è collocato un carattere in rilievo, che viene colpito da un martelletto non appena la margherita si è posizionata correttamente. Sostituendo quest'ultima, è possibile cambiare i caratteri di stampa. La qualità del risultato è di solito pari a quella di una macchina da scrivere ed è senz'altro superiore a quella di una stampante a matrice, anche multipass, benché la velocità sia relativamente bassa (di solito tra 8 e 70 cps).

Per ovviare alla lentezza, le più recenti stampanti a margherita sono bidirezionali, in modo da stampare sia durante il tragitto di "andata", sia durante quello di "ritorno". Inoltre, con speciali circuiti di ottimizzazione, si evita che la testina percorra inutilmente l'intera larghezza del foglio se non vi sono più caratteri da stampare.

Queste stampanti in genere offrono anche interessanti prestazioni addizionali: sono possibili neretti, sottolineature e tratti discendenti. Le margherite più recenti sono in materiale plastico: oltre a essere più leggere, i petali risultano maggiormente elastici, specie nel ritornare a posto dopo l'impatto. I caratteri sono generalmente realizzati in dura resina e, per una maggiore chiarezza di stampa, i bordi sono opportunamente affilati. Anche la densità di stampa può variare: quattro, cinque o sei caratteri per centimetro. Le margherite di ricambio costano più o meno quanto una cassetta di un gioco d'avventura.

Le stampanti a margherita sono insuperabili quanto a qualità di stampa. Per cambiare il set di caratteri basta sostituire la margherita (vedere il testo)

STAMPANTI SENZA IMPATTO

Meno diffuse di quelle a impatto sono le macchine a matrice di punti senza impatto, note anche come "termiche" o "elettrostatiche". Le stampanti termiche adoperano una speciale carta, trattata con una sostanza sensibile al calore. La carta è costosa, ma non richiede l'impiego di nastro inchiostrato. La testina contiene una matrice di elementi che, in una combinazione che dipende dalla forma del carattere, si scaldano tra i 100° e 150° centigradi, scurendo i pigmenti sensibili al calore della carta. Una delle stampanti di tipo termico più economiche è la ZX della Sinclair, compatibile sia con lo Spectrum che con lo ZX81. Costa circa un terzo dello stesso Spectrum.

Naturalmente, per una somma così modesta la stampante non offre grandi prestazioni: è però soddisfacente in molte applicazioni, nelle quali la qualità di stampa non è essenziale. Essa permette di stampare sia testi che grafica. I caratteri sono leggibili, anche se non sempre nitidi; ad ogni modo, serve egregiamente per ottenere una copia durevole di un listato o di un'immagine sullo schermo.

Oltre che per l'estrema economicità, la stampante ZX va ricordata per le sue ridottissime dimensioni: stà nel palmo di una mano e pesa meno di un chilo. Di conseguenza, però, il numero di caratteri per linea è limitato a 32 (quanti quelli dello schermo, del resto). La carta è disponibile in rotoli di 100 mm di larghezza, con una tipica lucentezza argentea dovuta al rivestimento in alluminio. Se si possiede uno ZX81 c'è lo svantaggio di dover ricorrere a un alimentatore più potente, poiché la stampante si alimenta direttamente dal computer.

Le stampanti elettrostatiche vanno rifornite periodicamente di *toner* (una sostanza solitamente composta da particelle di carbonio sospese in una soluzione di isoparaffina). I caratteri si formano caricando elettrostaticamente lo speciale rivestimento della carta e passando questa nel toner. Le particelle nere aderiscono alle zone caricate e in seguito, rimosso il toner in eccesso, la carta viene riscaldata per ottenere il *fissaggio* della stampa.

Le stampanti termiche ed elettrostatiche sono più affidabili di quelle a impatto, in quanto vi è una minore usura meccanica, ma hanno rispetto ad esse una risoluzione molto inferiore.

Una soluzione interessante è costituita dalle stampanti/plotter a penna di basso costo, nelle quali sono adoperate punte inchiostrate (talvolta normali penne biro o pennarelli) per tracciare caratteri preprogrammati oppure grafici e disegni. Esistono stampanti di questo tipo appositamente concepite per il Commodore.

In genere, sulla testina sono montate quattro penne di diverso colore e il normale set di caratteri alfanumerici è chiaro e leggibile. Con opportuni accorgimenti, comunque, si possono creare caratteri del tutto nuovi in forma e dimensione. La velocità di stampa si aggira sui 12 caratteri al secondo. Una serie completa di quattro penne di ricambio costa quanto un paio di cassette.

CONSIDERAZIONI SULLA CARTA

Nella scelta di una stampante, la cosa più importante è sapere quale deve essere l'effetto finale del prodotto, non solo per il tipo di carattere adoperato, ma anche per il formato e l'aspetto della carta. Eccezion fatta per le carte chimiche necessarie alle stampanti termiche ed elettrostatiche, esiste una grande varietà di moduli prestampati, con larghezze da 10 a 40 cm. Alcune stampanti possono usare tipi e formati diversi di carta, altre solo uno. La carta può essere acquistata in fogli

Le stampanti a penna trovano molte applicazioni nel disegno al computer e sono utili per generare un'ampia gamma di caratteri alfanumerici

sciolti, in rotoli o ripiegata. Quest'ultima, conosciuta anche col termine *fanfold*, è la tipica carta da stampante, perforata sui due lati e confezionata come un foglio continuo piegato a fisarmonica, con ogni pagina distinta da una dentellatura, per poter essere separata dalle altre.

A ogni tipo di carta corrisponde un meccanismo di rifornimento. Per la trazione della carta fanfold si usa un rocchetto dentato: i fori sui due lati dei fogli corrispondono ai denti sul rocchetto e ciò assicura un trascinamento perfetto. Con una stampante a impatto, si possono ottenere più copie semplicemente inserendo tra i fogli una carta carbone.

Per stampare su fogli sciolti o in rotoli serve un meccanismo di rifornimento a frizione (simile a quello delle macchine da scrivere). La carta è compressa da rulli tra due piani mobili. Il rotolo di carta si infila in un apposito sostegno situato nella parte posteriore della stampante, mentre per i fogli singoli l'inserimento è fatto manualmente. Questi ultimi sono senz'altro i migliori per la corrispondenza; gli altri vanno bene per listati o per la grafica.

INTERFACCIAMENTO

Prima dell'acquisto, c'è un'ultima considerazione da fare: la stampante deve essere compatibile, o poterlo diventare, con la propria marca di computer. Può essere necessario acquistare un circuito addizionale di interfaccia.

In sintesi, un'interfaccia è il dispositivo elettronico (insieme al software per controllarlo), che rende possibile collegare due sistemi. In certi casi basta un semplice cavetto, o perché la compatibilità è prevista in fase di costruzione, o perché l'interfaccia è incorporata. In altri, occorre acquistare del tutto l'interfaccia, il cavetto e i connettori adatti. Non si attenda fin dopo l'acquisto della stampante per verificare i requisiti dell'interfaccia: i costi addizionali potrebbero superare il prezzo dello stesso apparecchio!

Se entrambi gli apparecchi sono della stessa marca, di solito sono già compatibili o basta l'aggiunta di un'interfaccia. I produttori di stampanti non legati a specifiche marche di computer cercano di rendere i propri prodotti compatibili con il maggior numero possibile di computer, fornendo le interfacce adatte. Perciò, anche se l'interfacciamento è importante, il problema è relativo purché venga affrontato prima dell'acquisto.

Si possono acquistare moduli di conversione di interfaccia per alcuni apparecchi, ma può anche darsi che certi programmi che si intendono usare (specialmente i word processor) richiedano l'impiego di un'interfaccia particolare.

Per assicurare un certo margine di compatibilità tra prodotti di case diverse, esistono varie interfacce standard. La più famosa è la RS232C, un'interfaccia seriale in cui i dati di ogni carattere sono trasmessi e ricevuti sequenzialmente. Comunque, sia il computer che la stampante trattano l'informazione di ogni carattere in parallelo (ossia contemporaneamente) per cui è necessario convertire i segnali da paralleli a seriali e viceversa.

Questo è compito di un circuito integrato noto come UART (Ricevitore e Trasmettitore Universale Asincrono), che nel computer riceve i caratteri in codice o dalla tastiera o dalla memoria, li converte dal formato parallelo al seriale, aggiunge altri codici di trasmissione e, purché la stampante sia pronta, trasmette le stringhe di caratteri.

Un identico UART, nella stampante, riceve i dati, li decodifica e, se non ci sono errori, li emette in formato parallelo. La velocità di trasferimento dei dati varia tra le diverse stampanti da 75 a 19.200 bit per secondo, equivalenti a circa da 7 a 1800 caratteri per secondo.

Per evitare che il computer rimanga impegnato nella fase di stampa, rubando tempo prezioso alle elaborazioni, spesso la stampante viene corredata con un *buffer* di memoria, in cui vengono rapidamente scaricate le informazioni da stampare, consentendo al computer di proseguire con altre operazioni di calcolo. La stampante, nel frattempo, preleva e stampa i dati conservati nel buffer, le cui dimensioni vanno da 80 a 8000 caratteri. Maggiore è la memoria del buffer, più rapido è il trasferimento e maggiore è il tempo che si risparmia.

L'altro tipo di interfaccia seriale è quella "ad anello di corrente" (current loop) a 20 mA. Usata in origine sulle telescriventi, è oggi stata sostituita dalla molto più veloce RS232. Tra le interfacce da stampante parallele, la più diffusa è la Centronics: progettata dal costruttore della omonima stampante negli anni settanta, è diventata, specie per la sua semplicità, un vero standard. In questo caso, i dati vengono trasferiti, raggruppati in byte, su otto linee (una per bit) contemporaneamente.

Nato in origine per collegare strumenti di misurazione a un computer, lo IEEE 488 è un sofisticato standard di interfaccia parallela per comunicazioni in *duplex* (ossia in due sensi simultaneamente). Di conseguenza, essendo spesso presente sui microcomputer, questa interfaccia è adottata anche per le stampanti, benché la preferenza vada, come si è detto, alla Centronics.

Una volta deciso quale tipo di stampante acquistare, l'attenzione si rivolge ai prezzi. Si faccia attenzione a quelle macchine vendute senza cavi di connessione, che vengono poi a costare il doppio di un dischetto di giochi.

OLTRE LA BARRIERA DEL SUONO

Ravviviamo i giochi e rendiamoli più emozionanti con qualche effetto sonoro, dai bip alle esplosioni, dai colpi andati a segno alle marce funebri

Per accrescere l'emozione, i giochi possiedono effetti sonori di tutti i generi: esplosioni, sibili, rimbalzi e tutto ciò che la fantasia del programmatore sa creare. In questa lezione viene offerto un piccolo repertorio di effetti sonori prefabbricati, utilizzabili così come sono o come base per nuovi esperimenti.

Ricordiamo che non esistono regole inflessibili per produrre effetti sonori: se in un gioco serve un rumore di accompagnamento per un tizio che si stà buscando un fracco di botte, non resta che procedere per tentativi. Viceversa, alcuni suoni, apparentemente senza senso, diventano fantastici se abbinati alla grafica giusta.

Come incorporare gli effetti sonori nel programma dipende dalla loro complessità e dalla frequenza del loro impiego. Per effetti semplici, da usare una sola volta in un programma, può bastare una struttura del tipo IF ... THEN, ma per quelli più complessi o ricorrenti conviene scrivere apposite subroutine.

La raffinatezza degli effetti ottenibili dipende in parte dall'abilità del programmatore, ma anche dalle possibilità sonore del computer. Per esempio, il generatore di suoni dello Spectrum è limitato a un *bip*, di cui l'utente controlla tono e durata. Viceversa, i Commodore e gli Acorn sono dotati di generatori di suoni molto sofisticati, capaci di sintetizzare una vasta gamma di effetti diversi. Lo ZX81 non produce alcun suono.

Il BASIC dello Spectrum possiede un solo comando per il suono (BEEP), facile da adoperare e utile per vivacizzare i programmi. Ecco, per esempio, una routine per produrre tutta una sequenza di *bip*:

```
3000 FOR n=1 TO 12
3010 BEEP .03,30
3020 NEXT n
```

Come si può vedere da queste linee, BEEP è seguito da due numeri separati da una virgola.

Il primo numero determina la lunghezza della nota: più alto è il valore, più a lungo dura la nota. Con un valore 1, la nota dura un secondo, ma si possono usare valori maggiori oppure minori (frazioni decimali) di 1.

Il secondo numero assegna il tono alla nota: a Ø corrisponde DO centrale. Ogni numero intero superiore o inferiore a questo corrisponde a un semitono più alto o più basso, ossia un passaggio alla nota successiva sulla tastiera di un pianoforte. Per accompagnare esplosioni e spari nei giochi, si usano spesso toni molto bassi (con valore pari a circa −32).

L'uso del comando BEEP per la creazione di brani musicali verrà spiegato più approfonditamente in una successiva lezione. Nella routine appena vista, si usa un ciclo FOR ... NEXT per suonare una serie di

- INVENTARE IL SUONO GIUSTO
- AGGIUNGIAMO EFFETTI SONORI AL GIOCO DEL LABIRINTO
- L'USO DI **BEEP, PLAY, SOUND** ED **ENVELOPE**
- SONORIZZIAMO ESPLOSIONI, COLPI E SPARI
- DAL RUMORE ALLE NOTE E AI MOTIVI MUSICALI
- ALTRE ESPLOSIONI

note. Le seguenti linee adoperano due cicli. La variabile di controllo di ciascun ciclo stabilisce il tono di ogni nota, ottenendo un effetto adatto per un colpo andato a segno.

```
8000 FOR n=4 TO 0 STEP -1
8010 BEEP .01,n
8020 NEXT n
8030 FOR n=1 TO 4
8040 BEEP .01,n
8050 NEXT n
```

Ecco ora una routine con un ciclo FOR ... NEXT simile, ma con un effetto più adatto a un festeggiamento, utile in caso di eliminazione di tutti i mostri o in caso di tris giocando con una *fruit machine*.

```
8000 FOR n=10 TO 60 STEP 5
8010 BEEP .01,n
8015 BEEP .01,n-2
8020 NEXT n
```

Ci si può adesso esercitare con il comando BEEP per produrre altri effetti sonori per giochi, inserendoli nei programmi con cicli e subroutine.

SUONI PER IL LABIRINTO

Il gioco del labirinto casuale, visto nella precedente lezione, può essere migliorato a dismisura, con opportuni effetti sonori. Si aggiungano al programma originale le linee 365 e 500 e si cambi la 400:

```
365 BEEP .01,10
400 LET vite=vite-1:RESTORE 500:FOR f
    =0 TO 10: READ a,b: BEEPa,b:NEXT f
    : IF vite>0 THEN GOTO 260
500 DATA .45,0,.3,0,.15,0,.45,0,.3,3,.15,2,.3,2,
    .15,0,.3,0,.15,-1,.45,0
```

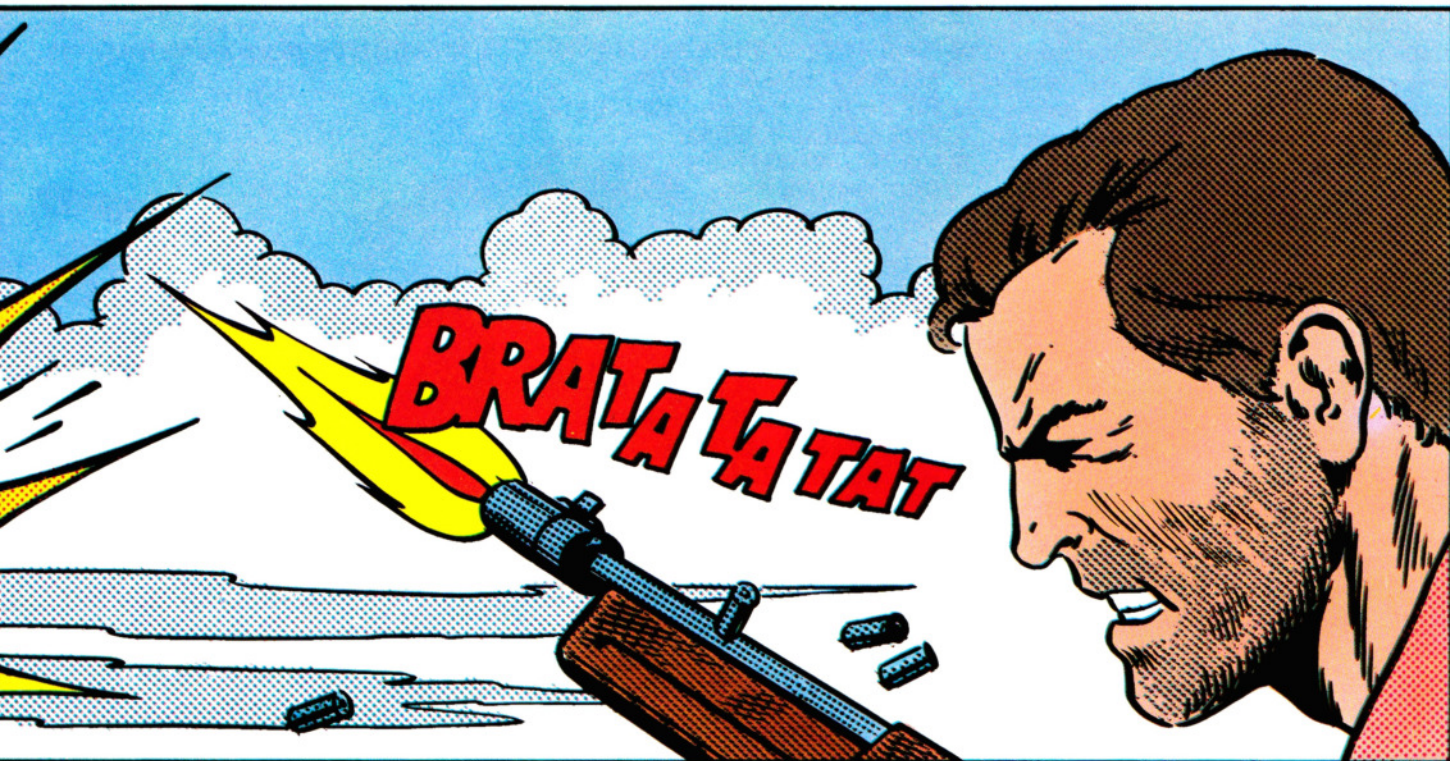
Adesso, lanciando il programma, si scoprirà che ogni vita perduta viene accompagnata da una marcia funebre.

Per creare effetti diversi si cambino i valori di a e b.

Questo esempio dimostra che occorre attenzione nell'introdurre effetti sonori nei giochi dato che, durante il loro svolgimento, ogni altra cosa si ferma e le pause possono riuscire sgradite. Anche se gli effetti sono di breve durata, il gioco potrebbe risultare troppo lento.

Il generatore di suoni del Commodore 64, tra i più sofisticati, ha tre canali o *voci* e consente l'emissione di una grande varietà di suoni e rumori, adattissimi a commentare i più svariati giochi.

Come vedremo tra breve, le varie voci possono essere combinate assieme, ma una spiegazione approfondita su come programmare efficacemente il chip SID verrà approfondita in una prossima lezione della sezione "Programmazione BASIC". Il chip SID è estremamente versatile e con un relativamente piccolo intervento di programmazione si possono creare numerosi e utili effetti sonori. A differenza del chip VIC per la grafica, che richiede un uso esteso di POKE più volte nel programma, per il chip SID la maggior parte delle POKE necessarie vanno eseguite una sola volta.

In seguito, bastano piccoli cambiamenti nei valori per modificare radicalmente la natura del suono prodotto.

Per ogni suono, in genere occorrono fino a cinque tipi di assegnazione; al fine di valutarne gli effetti, si trascrivano queste linee:

```
1Ø S=54272
3Ø POKE S+1,255
4Ø POKE S+5,219
5Ø POKE S+24,15
6Ø POKE S+4,129
7Ø FOR Z=1 TO 5ØØØ: NEXT Z
8Ø POKE S+4,128
1ØØ GOTO 3Ø
```

Eseguendo il programma, si dovrebbe udire un suono di onde che si frangono sulla riva. Si noti come le locazioni del chip del suono vengano indicate aggiungendo un particolare valore all'indirizzo di base S, ossia 54272: ciò rende molto più facile ricordarsi a quale locazione si riferisce ciascun comando. La prima assegnazione, S+1, riguarda uno dei due possibili registri di controllo della *frequenza*: qui viene usato quello che influenza il byte "alto", ottenendo cioè un valore di frequenza 255 volte il numero specificato, che nell'esempio in questione, è il massimo possibile: 255.

Per ottenere un rumore di onda diverso, si provi a diminuire questo valore (ad esempio portandolo a 1Ø) e si riesegua il programma.

La locazione S+5 corrisponde al *generatore di inviluppo* del quale non ci occuperemo per il momento.

Nella linea successiva, S+24 è la locazione del *controllo di volume* quando si usano valori inferiori a 15, che corrisponde al massimo volume. Questo è il valore impiegato nel nostro esempio (nel caso in cui fosse troppo alto, basterebbe regolare il comando sull'apparecchio TV).

Alla linea successiva si usa S+4, la locazione del registro di controllo della voce−1, usato per selezionare, tra le altre cose, la *forma d'onda* (assegnando opportuni valori ai bit più "alti").

Il valore 129 seleziona la forma d'onda di un rumore casuale: per alzarne il tono si usi 33 o 17.

Per prova, si lasci il valore 17, si cambi il valore della POKE alla linea 3Ø in 1Ø e si abbrevi il ciclo per la pausa alla linea 7Ø sostituendo 5Ø a 5ØØØ. Eseguendo il programma si sentirà che il rumore del mare si è trasformato in quello di un treno a vapore!

In un programma definitivo, simili cambiamenti di suono si possono introdurre facilmente se nelle POKE, anziché adoperare delle costanti, si usano delle variabili il cui contenuto viene man mano variato dallo stesso programma.

SUONI PER IL LABIRINTO

Se si è conservato il gioco del labirinto presentato a pagina 196, possiamo ora rileggerlo in memoria e aggiungere le tre linee seguenti (prima ricordiamoci di spegnere e riaccendere il Commodore):

```
1Ø2 S=54272:POKE S+5,33:POKE S+
    6,255:POKE S+24,15:POKE S+4,33:
    POKE S+1,Ø
2ØØ2 POKE S+1,Z
3ØØ5 POKE S+1,25Ø:FOR Z=1 TO 1Ø:
    NEXT Z:POKE S+1,Ø
```

La prima linea inizializza il sistema audio, la seconda e la terza aggiungono gli effetti sonori adatti a rendere più interessante il gioco. Ecco ora alcuni altri effetti con cui fare esercizio. Si comincia predisponendo il sistema sonoro (prima si esegua NEW):

```
5 S=54272:W(1)=17:W(2)=33:W(3)=129
1Ø FOR Z=S TO S+24: POKE Z,Ø: NEXT Z
15 P=1 :W=1 :REM P (1−13)W (1−3)
2Ø POKE S+24, 15 :REM VOLUME
25 POKE S+5,15 :REM ATTACCO/
    DECADIMENTO
3Ø POKE S+4,W(W):REM FORMA D'ONDA
35 POKE S+6,15 :REM PERMANENZA/
    RILASCIO
4Ø ON P GOSUB 55,6Ø,65,7Ø,75,8Ø,85,9Ø,95,
    1ØØ,1Ø5,11Ø,115
5Ø POKE S+4,W(W)−1:POKE S+5,Ø
    : POKE S,Ø:POKE S+1,Ø:END
55 FOR Z=1 TO 75 STEP.1: POKE S+1,Z
    : NEXT Z: RETURN
6Ø FOR Z=1 TO 75 STEP.1: POKE S+1,
    ABS(SIN(Z)*15):NEXT Z: RETURN
65 FOR Z=75 TO 5 STEP−1:POKE S+1,Z:
    POKE S,Z:NEXT Z: RETURN
7Ø FOR Z=1 TO 1ØØ:POKE S+1,RND(1)*75:
    NEXT Z:RETURN
75 FOR Z=1 TO 2ØØ:POKE S+1,ABS(TAN
    (Z)+5):NEXT Z: RETURN
8Ø POKE S+1,1Ø:POKE S,127:FOR Z=
    1 TO 15 STEP.Ø5:POKE S+24,Z:NEXT Z:
    RETURN
85 FOR Z=1 TO 25Ø STEP 1:POKE S+1,Z:
    POKE S+1,255−Z:NEXT Z:RETURN
9Ø FOR Z=5 TO 1ØØSTEP 5: FOR ZZ=
    1Ø TO 1ØØ STEP 1Ø:POKE S+1,Z:
    POKE S,ZZ:NEXT ZZ,Z: RETURN
95 FOR Z=2Ø TO 2ØØ STEP 1Ø: FORZZ=
    1 TO 2Ø:POKE S+1,Z−ZZ:POKE S+
    1,ZZ+5Ø:NEXT ZZ,Z: RETURN
1ØØ FOR Z=1 TO 4Ø:FOR ZZ=1 TO RND(1)
    *5Ø:POKE S+1,ZZ:NEXT ZZ,Z:RETURN
1Ø5 FOR Z=1Ø TO 1ØØ STEP 1Ø:FOR ZZ=
    5 TO Z:POKE S+1,Z−ZZ:NEXT ZZ,Z:
    RETURN
11Ø POKE S+1,RND(1)*2ØØ+5:FORZ=
    1 TO RND(1)*5ØØ+1ØØ:NEXT Z:RETURN
115 POKE S+1,RND(1)*2ØØ+5:FORZ=
    1 TO RND(1)*1ØØ STEP .2:POKE S+24,Z:
    NEXT Z:RETURN
```

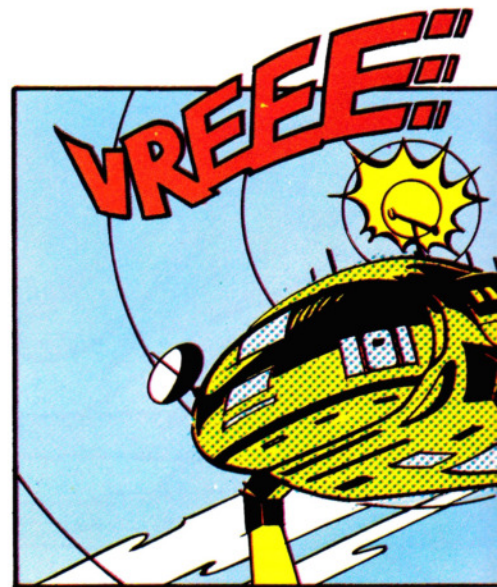
Le linee da 1Ø a 5Ø inizializzano il sistema e ogni linea successiva riguarda un effetto sonoro a sé, da introdurre nel programma adattando adeguatamente i valori P e W alla linea 15. P rappresenta gli effetti contenuti in una linea sola dalla linea 55 alla 115. W stabilisce la forma d'onda e, variandone il valore da 1 a 3, si seleziona tra le forme *triangolare, a dente di sega* o *rumore* quella desiderata per gli effetti.

L'esecuzione del programma così com'è produce un alto grido. Si ripulisca lo schermo, si chieda LIST della linea 15 e si riesegua il RUN.

Per provare le tre variazioni dei suoni prodotti da ciascuna linea, possiamo modificare P e W.

Per esempio, con P=2 W=1 si ha un gorgheggio, che diventa un barrito se poniamo W=3. Proviamo anche le combinazioni P=7 W=1, P=9 W=1, P=1Ø W=1 o 2, P=12 W=1 e tutte quelle che ci vengono in mente.

Altri effetti si ottengono alterando il valore delle POKE alle linee 25 e 35.

La generazione di suoni sul Vic 2Ø è molto simile a quella del C64, ma meno sofisticata. Cinque registri regolano la produzione dei suoni: la locazione 36874 per le note basse, la 36875 per le note medie e la

36876 per le alte. La 36877 si riferisce a un generatore di rumore e la 36878 controlla il volume. Normalmente, questi registri contengono il valore Ø (off). I registri del tono possono contenere valori da 128 a 255 e il registro del volume da 1 a 15. Se, in uno dei registri del tono, si deposita un numero maggiore di Ø, vengono prodotti suoni, fino la loro spegnimento, che si ottiene depositandovi nuovamente il valore Ø con una POKE.

Ecco un piccolo repertorio di suoni per illustrare l'uso dei registri nella produzione di effetti speciali. Il primo è il suono per un 'finale' di gioco:

```
10 POKE 36878,15
20 FOR Z=128 TO 255:POKE 36875,Z:
   NEXT Z
30 POKE 36875,Ø
```

ecco una motocicletta:

```
10 POKE 36878,15:FOR Z=1 TO 100
20 POKE 36875,(150+RND(1)*5)
30 POKE 36875,Ø:NEXT Z
```

Questo è un interessante effetto fantascientifico, che prende in prestito valori da altre locazioni di memoria:

```
10 POKE 36878,15
20 FOR Z=60000 TO 60999: POKE 36876,
   PEEK(Z):NEXT
30 POKE 36876,Ø
```

Gli apparecchi Acorn possiedono due comandi per produrre suoni: SOUND e ENVELOPE. SOUND, da solo, produce un effetto semplice, mentre ENVELOPE apporta modifiche alla forma del suono, per ottenere effetti più ricercati.

Ecco il tipo di effetto ottenibile con il solo SOUND:

```
SOUND 1, -15,80,100
```

I quattro numeri selezionano: il canale, il volume, il tono e la durata della nota. Tra i due canali possibili, Ø e 1, il primo serve per i toni musicali puri, il secondo per il rumore.

I valori possibili per il volume vanno da −15 (alto) a Ø (spento). I numeri tra 1 e 16 sono i numeri relativi al comando d'inviluppo, ENVELOPE, come si vedrà poi.

Il valore del tono può variare da Ø a 255; Ø produce una nota molto bassa e 255 una molto acuta.

Infine si può variare la durata tra Ø e 255, maggiore è il numero, maggiore è la durata: 255 dà una nota continua, che dura finché non si preme ESCAPE, qualora non provveda il programma a terminare l'emissione.

Uno dei maggiori pregi della generazione di suoni sugli Acorn è che essa non distoglie il computer dalla prosecuzione del proprio lavoro durante l'emissione di effetti audio. In altri apparecchi invece si creano fastidiose pause: il computer deve sospendere ad esempio la creazione di un immagine grafica, se c'è da suonare una musichetta.

Quanto detto finora vale sia per il BBC che per l'Electron, sebbene esistano delle differenze tra i due apparecchi: il BBC è più "dotato musicalmente".

In primo luogo, il BBC possiede due canali in più, il 2 ed il 3, identici all'1 e usati per produrre note musicali in modo polifonico (più note emesse contemporaneamente). In secondo lugo, il volume aumenta gradualmente da Ø a −15. L'Electron accetta valori intermedi, ma ogni numero minore di Ø corrisponde a 'acceso'.

SUONI PER IL LABIRINTO

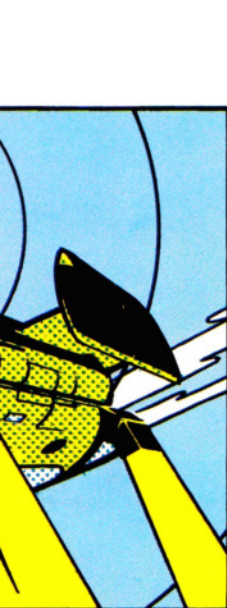
Se si è conservato il gioco del labirinto di pagina 197, possiamo adesso aggiungervi alcuni effetti sonori. Ecco intanto un suono per accompagnare il movimento attraverso il labirinto:

```
795 SOUND1,-15,150,1
```

Il secondo suono subentra quando il giocatore trova un tesoro; si aggiunga la linea 225 e si cambi la 83Ø:

```
225 ENVELOPE1,128,2,Ø,Ø,50,Ø,Ø,127,Ø,Ø,
    -127,126,Ø
830 IF H=226 THEN SOUND 1,1,150,10:
    PRINT TAB(RND(36+D*40)+1,RND(22)
    +4)CHR$(226): SC=SC+MT-TIME
    : TIME=Ø
```

Per finire, una marcia funebre per quando si perde una vita.

```
891 RESTORE: FOR T = 1 TO 11
892 READ K,P
893 SOUND1,-15,P,K
894 SOUND1,Ø,Ø,1
895 NEXT
896 DATA 12,5,8,5,4,5,12,5,8,17,4,13,8,13,4,
    5,8,5,4,1,12,5
```

ENVELOPE

Il suono per la conquista del tesoro usa le istruzioni ENVELOPE e SOUND. ENVELOPE serve per modificare quanto prodotto da SOUND.

L'uso del comando ENVELOPE, che prevede ben 14 parametri, è piuttosto complesso e verrà trattato a fondo in una successiva lezione. Comunque, un'idea sul funzionamento la si può già acquisire esercitandosi con gli ENVELOPE qui presentati.

Se si decide di usare la combinazione di comandi ENVELOPE e SOUND, occorre prestare attenzione affinché il secondo valore nel comando SOUND (che corrisponde al volume), coincida col primo valore di ENVELOPE.

ALTRI EFFETTI SONORI

Ecco una serie di effetti sonori da inserire in programmi di giochi. La linea ENVELOPE può essere inserita dovunque nel programma, purché preceda il comando SOUND a cui si riferisce. In molti giochi è spesso utile un'esplosione come questa:

```
10 ENVELOPE1,131,0,0,0,0,0,0,126,–3,0,0,
   126,0
20 SOUND 0,1,6,100
```

Come alternativa, possiamo ottenere un fracasso come questo:

```
10 ENVELOPE1,1,0,0,0,0,0,0,126,–1,–3,
   –126,126,0
20 SOUND 0,1,60,35
```

Oppure una sirena come questa:

```
10 ENVELOPE3,1,3,–3,3,10,20,10,127,0,0,0,
   126,126
20 SOUND 1,3,100,20
```

Simili suoni sarebbero adattissimi a un gioco non del tipo "spaziale", ossia con carri armati o con auto in corsa.

Per un gioco di caccia all'alieno, servono suoni un po' più astratti come questo:

```
10 ENVELOPE1,130,20,–2,0,5,30,0,126,0,0,
   –7,126,0
20 SOUND 1,1,100,10
```

O rumori secchi:

```
10 ENVELOPE2,1,0,0,0,0,0,0,126,–1,–1,
   –1,126,0
20 SOUND 1,2,170,3
```

O un raggio vibrante:

```
10 ENVELOPE1,1,–16,48,–16,3,1,3,7,65,0,
   –4,65,126
20 SOUND 1,1,220,8
```

Si possono usare questi effetti come sono oppure abbinare a SOUND un ENVELOPE diverso, ricordandosi sempre di far corrispondere i valori di ENVELOPE. Inoltre, possiamo esercitarci con profitto, cambiando a turno anche qualche altro valore nei due comandi.

T

Due sono i comandi BASIC che producono effetti sonori sul Dragon e sul Tandy: SOUND e PLAY. Entrambi sfruttano lo stesso generatore di suono e la qualità del suono prodotto non può essere modificata di molto. SOUND controlla il tono e la durata della nota, mentre PLAY consente di specificare una stringa di note per modulare, ad esempio, un motivetto musicale.

SOUND è il più semplice da usare e produce effetti come quello del seguente programma (prima di impartire il RUN si regoli il volume della TV):

```
10 FOR Q=1 TO 5
20 SOUND 200,1
30 NEXT Q
```

La serie di *blip* può essere modificata, variando i valori nel comando SOUND: il primo controlla il tono della nota e può avere un valore da 1 a 255. 1 produce la nota più bassa e 255 la più alta. Come orientamento, per chi conosca un po' di musica, 89 corrisponde al DO centrale.

Il secondo numero dopo SOUND regola la durata dei *blip*, ancora secondo un arco di valori da 1 a 255: 16 dà una durata di circa un secondo.

Questi suoni vanno bene in un gioco spaziale o di fantasia, ma non in uno di tipo più realistico. Come accennato già a pagina 161, è necessario prestare attenzione a dove, nel programma, si inseriscono gli effetti e ad avviare la generazione di un effetto adatto al gioco.

Se però vogliamo adottare effetti più sofisticati, si abbandoni l'uso del troppo limitato comando SOUND e si adoperi invece PLAY, come esposto di seguito.

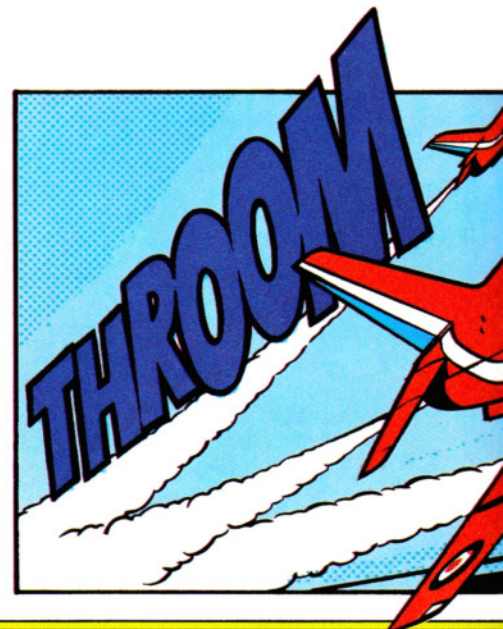
SUONI PER IL LABIRINTO

Se si è conservato il gioco del labirinto esposto alle pagine 194-196, ora lo si può migliorare aggiungendo questi effetti.

Si cambi la linea 230 in:

```
230 IF X<>LX OR Y<>LY THENPUT(X1,
   Y1)–(X1+BS–1,Y1+BS–1),B,PSET:
   LX=X:LY=Y:PLAY"T5005DG"
```

e la linea 240 in:

```
240 IFF=1 THENF=0:SC=SC+(TI–
   TIMER):TI=TI–10:PLAY"T1003BCDEFG
   A":GOTO130
```

e infine si cambi la linea 500 in:

```
500 CLS:SCREEN0,0:PLAY"T302L2CL4CL12
    CL2CL4D#L8DL4DL8CL4C01L8B02L2C"
    :LI=LI-1
```

Si possono correggere le linee esistenti o riscriverle completamente. Ora si esegua un RUN e si ascolti cosa succede spostando un personaggio. Quando si raggiunge il tesoro è emesso un suono diverso e quando si perde una vita suona una marcia funebre.

Va prestata attenzione nell'inserire simili commenti sonori in un gioco, per evitare che le pause introdotte da queste routine non interrompano troppo l'andamento del gioco. Infatti, ogni volta che il computer emette suoni, viene sospesa ogni altra attività e un gioco può risentirne in modo determinante, tant'è vero che talvolta, quando si introducono volutamente delle soste, si usa ricorrere all'emissione di un qualche motivetto. Ecco perché il suono generato alla linea 230 per mantenere la pausa al minimo è di brevissima durata.

IL COMANDO PLAY

PLAY opera sempre su una stringa contenente istruzioni per il computer. La stringa alla linea 230 del programma precedente contiene le istruzioni T5005DG. T stà per Tempo (in senso musicale), e può assumere ogni valore da 1 a 255. O si riferisce all'Ottava e può variare da 1 a 5 (1 la più bassa, 5 la più alta). Le ultime due lettere, D e G, rappresentano le note Re e Sol nella notazione anglosassone. Insomma, T5005DG sta per 'suonare due note alte abbastanza rapidamente'.

Si provi a cambiare i valori di T e di O per sentire che effetti si ottengono.

Alla linea 240 la stringa è T1003BCDEF GA, che suona una scala ascendente partendo dalla nota Si.

La marcia funebre, alla linea 500, è più complessa. Una spiegazione esauriente su come scrivere musica col computer verrà data in una successiva lezione: intanto si noti come la lunghezza della nota, L, vari durante la canzoncina. Si possono cambiare tutti i parametri nella stringa in qualsiasi momento del motivo, rendendo molto elastico l'uso di PLAY.

ESPLOSIONI

Il comando PLAY è utile anche per produrre il suono di esplosioni, che ricorrono spesso nei giochi.

I seguenti effetti possono essere usati da soli o assieme agli effetti presentati nelle pagine 161-167:

```
10 PLAY "T16001L30GF#FE#D#DC#D
   #DFE#"
```

Questo suono è utile quando qualcosa è colpito o esplode sullo schermo. Per esempio, lo si può inserire nel programma a pagina 100, aggiungendo il comando PLAY alla linea 270, prima di GOTO.

Un piacevole suono *riverberato* si ottiene così:

```
10 PLAY "T8001L2BFBFBFBFBFBFBFBFBF"
```

Due note vengono ripetute più volte, nell'ottava più bassa. Si intervenga sul tempo e sulla durata per modificare questo suono. La ripetizione di una sequenza di note si può ottenere con una routine di questo tipo:

```
10 F$="T10002L10AGBE#DFADF#"
20 F$=F$+F$
30 PLAY F$
```

La linea 10 è la stringa di istruzioni per il suono, la linea 20 concatena la stringa cosiché, quando la linea 30 suona, il suono si ripete.

UNA SIRENA

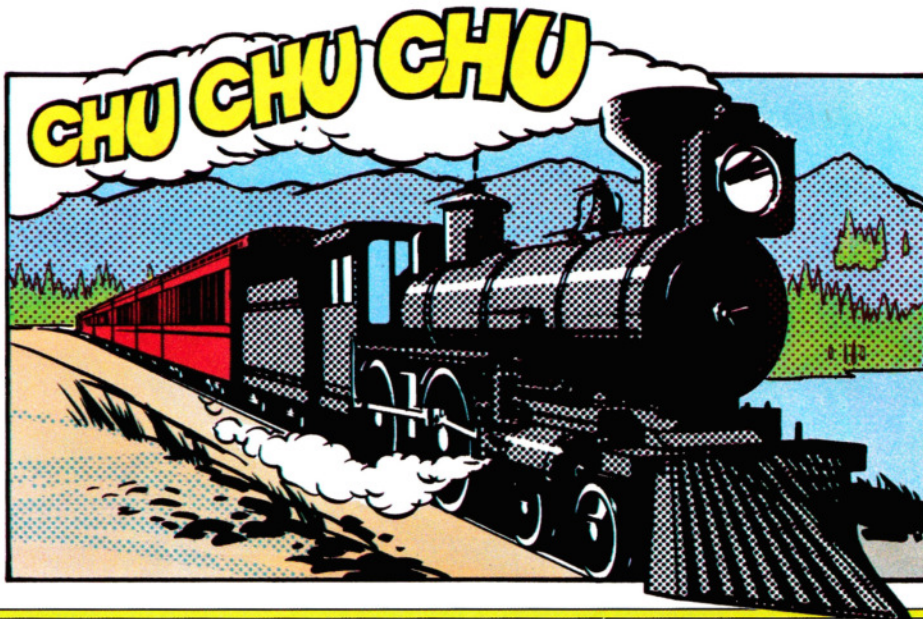
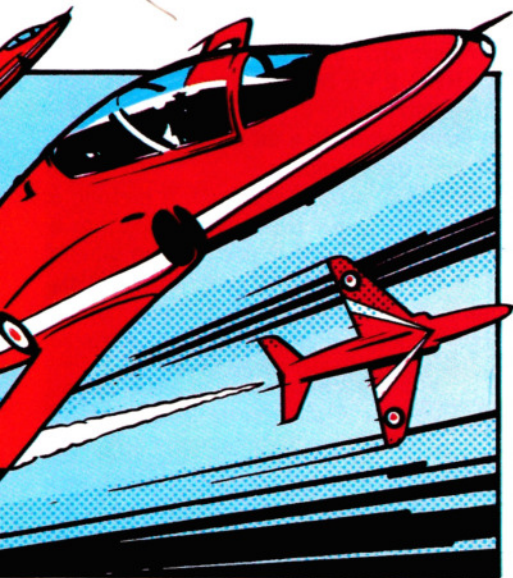
Sapendo come ripetere un suono, si possono trovare varie applicazioni. Una sirena può avvertire un giocatore di un nemico in arrivo. Si può usare un programma come questo:

```
10 CLEAR 250
20 F$="T15004L8CDD#EF#FGF#FED#
   DC#CD#DC#C"
30 F$=F$+F$+F$+F$
40 PLAY F$
```

Per prolungare l'effetto sirena basta concatenare più volte la stringa originale F$, prima di eseguirla. Tuttavia, il Dragon e il Tandy, all'accensione hanno soltanto una limitata quantità di memoria riservate alle stringhe, non sufficiente per sequenze molto lunghe. A ciò si ovvia riservando un maggior spazio alle stringhe: la linea 10 si occupa proprio di questo, riservando 250 byte. Si ricordi di accantonare questi 250 byte di memoria, quando si prevedono stringhe di notevole lunghezza in un gioco, altrimenti il computer segnala un errore OS (spazio delle stringhe esaurito).

Non ci si scoraggi se non si ottegnono effetti paragonabili a quelli dei giochi in commercio: in BASIC si producono buoni effetti, ma i più raffinati sono scritti in codice macchina, come verrà spiegato in una prossima lezione.

MIRIAMO AL CUORE

Capire il codice macchina implica capire anche come funziona e cosa è capace di fare il microprocessore, cervello e cuore della macchina

I computer hanno memorie, ma non servono soltanto per ricordare informazioni: sono anche capaci di elaborare ciò che memorizzano. La manipolazione è opera dell'Unità Centrale di Processo (CPU), ossia il microprocessore.

Un microprocessore sembra un chip come tanti altri, ma in realtà è ben diverso dai dispositivi passivi, capaci solo di ricordare quanto in essi inserito. Il microprocessore è *intelligente*: sa sommare, sottrarre, muovere un'informazione da una locazione di memoria all'altra, scorrere e ruotare le configurazioni di bit. Imparare a comunicare col microprocessore e saperlo sfruttare appieno è la sostanza del codice macchina.

IL CUORE DELLA MACCHINA

Tutto ciò che accade nel computer è sotto il diretto controllo del microprocessore, che esegue calcoli, trascrive dati da una locazione di memoria all'altra, assegna funzioni ad aree di memoria ed esegue i programmi. Scrivere programmi in codice macchina significa dunque rivolgersi direttamente al microprocessore. Al suo interno ci sono alcune locazioni di memoria speciali, dette *registri*, ognuno dei quali ha una funzione specifica. Quali di questi registri vengano impiegati dipende dalle istruzioni del codice macchina che compongono i programmi. Una loro caratteristica è di non avere indirizzi, il che li rende rapidi da utilizzare.

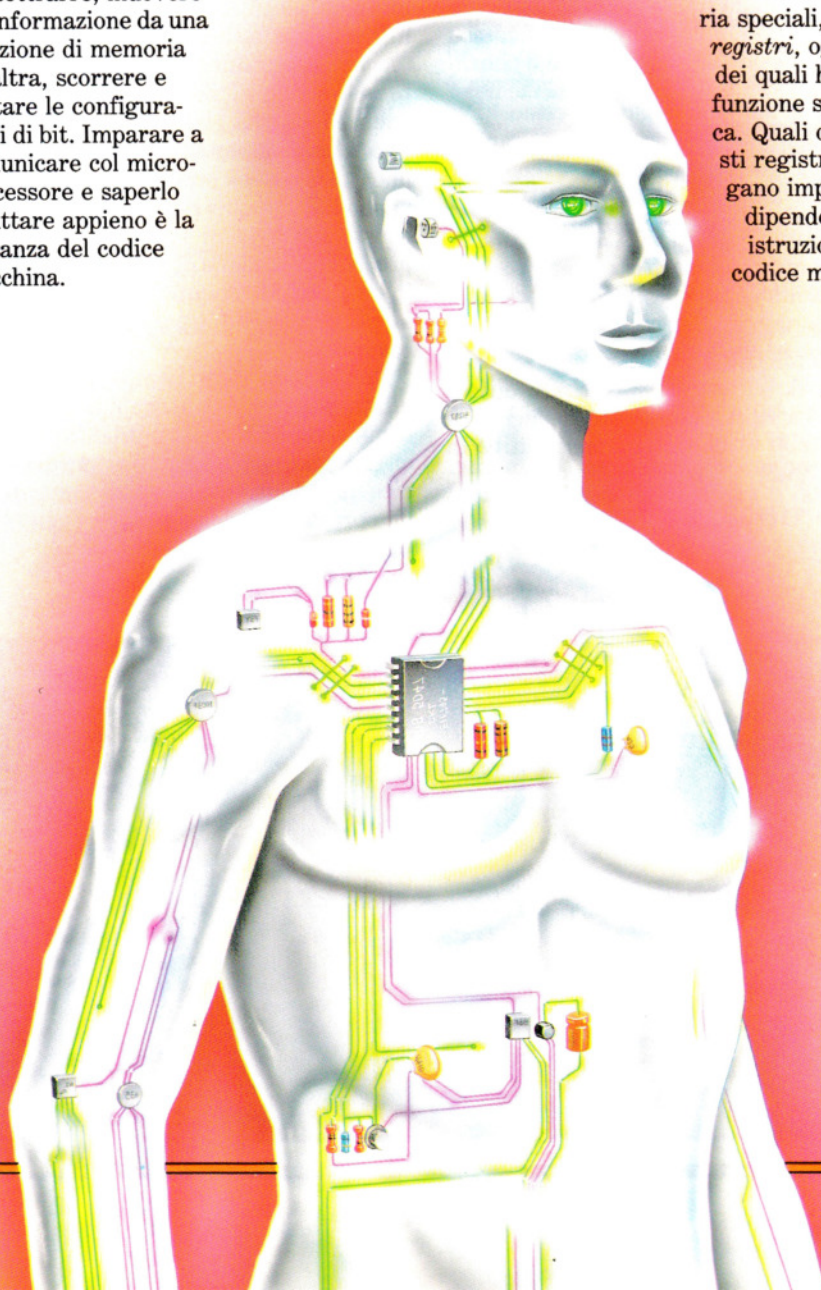
I REGISTRI

Ognuno dei vari chip che stanno alla base degli home computer trattati da INPUT ha una diversa configurazione di registri; ciononostante, vi sono elementi comuni a tutti gli apparecchi. Tutti hanno un registro di 16 bit, detto *contatore di programma* (più brevemente PC), che contiene l'indirizzo del prossimo byte del programma in codice macchina da eseguire. In altre parole, il PC indica al computer il percorso da seguire. Ogni volta che viene eseguita un'istruzione, il registro PC viene automaticamente incrementato per puntare a quella successiva.

Tutti gli home computer hanno almeno un *accumulatore* di otto bit (il registro A), nel quale il computer fa i calcoli aritmetici. Per l'addizione o la sottrazione di due numeri, uno di essi deve trovarsi nell'accumulatore. Ciò vale anche per la moltiplicazione e la divisione, che si riducono a combinazioni di operazioni logiche con addizioni e sottrazioni. Inoltre, esiste anche un paio di *registri indice*. Nello Spectrum, nello ZX81, nel Tandy e nel Dragon questi sono registri di 16 bit, che contengono indirizzi i quali possono essere incrementati o decrementati o sottoposti a operazioni aritmetiche nell'accumulatore, per dare altri indirizzi. L'Electron, il Micro BBC, il Vic 2Ø e il Commodore 64 hanno registri indice di otto bit il cui contenuto viene sommato a un indirizzo per ottenere un nuovo indirizzo.

Qualsiasi criterio si adotti, il metodo è noto come *indirizzamento indicizzato* e viene per lo più usato per leggere tabelle di dati in memoria: il programma inizia dall'indirizzo del primo byte della tabella e poi procede incrementando il registro indice a ogni passo successivo.

I FLAG

In ognuno dei nostri home computer, un registro di otto bit contiene i cosiddetti *flag*, ossia singoli bit il cui valore (1 o Ø) dipende dall'esito di certe operazioni, effettuate in genere nell'accumulatore.

- COS'È E COSA FA LA CPU
- COMUNICARE CON IL MICROPROCESSORE
- I REGISTRI
- COME IL COMPUTER ESEGUE I PROGRAMMI
- COSA INDICANO I FLAG
- LE LOCAZIONI DI MEMORIA E LO STACK

Come si conta in 'computerese'

Su un computer il conteggio parte sempre da zero. Ad esempio, le locazioni di memoria si numerano da ØØØØ a FFFF. Lo stesso accade con i bit in un byte: si contano da destra a sinistra, partendo da zero.

Cosicché, la cifra binaria all'estrema destra viene chiamata bit Ø. Il bit adiacente, a sinistra del bit Ø, è chiamato bit 1 e via dicendo.

In alcuni casi, termini come 'a destra', o 'a sinistra' sono ambigui, specie con i numeri di due byte memorizzabili nei due sensi a seconda del modo di operare del computer; In 'computerese' esiste la distinzione tra il byte "più significativo" (quello di valore più alto) e quello "meno significativo". Perciò, se si sta memorizzando un indirizzo, ad esempio 3D8E, allora 3D è il byte più significativo, mentre 8E è il meno significativo.

I Sinclair, i Commodore e gli Acorn memorizzano il byte basso o meno significativo (qui 8E), nella locazione di memoria più bassa. Il byte alto (3D in questo caso), va invece nella locazione di memoria con indirizzo immediatamente superiore, cioè nell'indirizzo che contiene 8E, più 1. C'è una sola eccezione a questa convenzione basso-alto: i numeri di linea nel BASIC vengono memorizzati in alto-basso.

Il Dragon e il Tandy memorizzano tutti i loro numeri di due byte con il byte alto nella locazione più bassa e il byte basso nella locazione più alta.

La stessa terminologia vale per i bit. Nel numero binario Ø1Ø1Ø1Ø1 il bit più significativo, il bit sette, vale Ø e il meno significativo, il bit zero, vale 1.

Ogni flag ha un preciso significato. Ad esempio, il bit meno *significativo* del registro, ossia quello situato più a destra, è il *carry* (riporto) o anche flag C. A seguito di una qualsiasi operazione aritmetica (somma, sottrazione, ecc.) o logica (AND, OR, ecc.), che richieda un riporto o un prestito di bit, il flag C viene automaticamente posto a 1.

Altri bit del registro indicano se il risultato di una certa operazione è zero (il flag *zero*), o negativo (il flag *segno*), o se supera la capacità registro (il flag *overflow*). Anche se comuni a tutti gli apparecchi, questi flag non occupano necessariamente la stessa posizione nel registro: alcuni flag, inoltre, sono specifici di ogni apparecchio.

I flag sono indispensabili per creare strutture condizionali, simili alla IF ... THEN del BASIC. Nel codice macchina esistono istruzioni di "salto", che deviano il percorso del programma a seconda del valore indicato da uno dei flag.

LO STACK

Un altro registro, comune a tutti i computer, è il *stack pointer* (puntatore allo stack, o più brevemente SP), che contiene l'indirizzo relativo a una particolare zona di memoria, chiamata appunto *stack* della macchina. Quest'area di memoria, rapidamente accessibile, viene adoperata per appoggiarvi temporaneamente indirizzi e dati, senza dover ricorrere alle normali procedure d'indirizzamento.

Lo stack è paragonabile a una pila di appunti: quando aggiungiamo un'informazione, il foglietto viene appoggiato in cima alla pila e, se un foglietto va tolto, occorre farlo sempre dalla cima: l'ultimo a entrare è il primo a uscire.

Il microprocessore deposita dati in cima allo stack e da esso, quando gli viene chiesto, riprende il primo elemento. Ciò evita di dover specificare la locazione di memoria relativa al dato o la sua posizione nello stack: ogni volta non c'è che una locazione di cima, per cui il computer non può sbagliare.

Al contrario, il programmatore può facilmente confondersi e dimenticare l'esatta posizione di un'informazione nello stack e recuperare perciò dalla cima un dato per un altro. Occorre dunque la massima attenzione. Appena afferrato il concetto di stack, come pila sulla cui cima depositare e recuperare informazioni, ecco subito un fatto che ci confonde: in tutti i computer qui considerati, la base dello stack è

situata verso la cima della RAM e il resto dello stack, elemento per elemento, cresce verso il basso. In pratica quindi (ma si tratta pur sempre si convenzioni), i dati vengono depositati e ripresi dal *fondo* dello stack e non dalla cima.

Ciò significa che, inserendo un nuovo elemento nello stack, automaticamente il puntatore decrementa, mentre quando le informazioni vengono tolte esso incrementa. Ciononostante, l'esempio della pila rimane valido, con la base dello stack fissata da una variabile di sistema e l'altra estremità libera per accogliere uno a uno i vari elementi.

SUBROUTINE

Il computer usa lo stack per conservare l'indirizzo dell'istruzione cui ritornare, quando nel programma incontra la chiamata a una subroutine: in questo caso il contenuto del contatore di programma viene depositato in cima allo stack e l'indirizzo della subroutine viene trasferito nel contatore di programma.

Eseguite tutte le istruzioni della subroutine, il contatore di programma viene regolarmente incrementato incontrando l'istruzione che impone al computer di tornare al programma "chiamante". Questa istruzione fa sì che l'indirizzo in cima allo stack venga trasferito nel contatore di programma, che viene incrementato nel modo abituale. Ecco perché, nei programmi in BASIC, ogni routine nidificata deve essere completamente contenuta nella precedente, altrimenti si confondono gli indirizzi contenuti nello stack.

Ora che si possiedono i concetti più importanti della programmazione in codice macchina, vediamo come metterli in relazione ai chip dei vari computer.

Nel chip Z8Ø dello Spectrum o dello ZX81 ci sono 21 *registri per 'utente'*, che il programmatore ha la facoltà di controllare.

Esistono due accumulatori di otto bit, A e A', non utilizzabili contemporaneamente, ma commutabili per eseguire due operazioni simultanee. È meglio non provarci con lo ZX81, altrimenti, nel modo SLOW, si perderebbe l'immagine.

Il registro dei flag, F, è di otto bit, ma non tutti questi riguardano il programmatore. Il bit Ø è il flag C di riporto, il bit 1 è il flag N, sottrazione, che interviene solo quando si usano numeri espressi in una speciale forma *decimale codificata in binario*. Il bit 2 è il flag P/V di parità/traboccamento, il bit 4 è il flag H, mezzo riporto, anch'esso usato nel decimale in codice binario, il bit 6 è il flag Z di zero e il bit 7, infine, è il flag S di segno. Associato al registro alternativo A' esiste un registro di flag F', di pari significato.

Esistono poi sei registri per uso generico: BC, DE e HL, utilizzabili separatamente come registri di otto bit o a coppie come registri di sedici bit. C' è anche una serie alternativa: B'C', D'E' e H'L'. I registri HL e H'L' si possono usare anche come accumulatori di 16 bit. Tuttavia, il registro H'L' va riportato al suo valore originale prima di tornare in BASIC.

Il puntatore dello stack è a 16 bit, come i registri indice IX e IY. Questi ultimi contengono indirizzi da incrementare o da sfasare sommando o sottraendo un valore chiamato *offset* (scarto), onde ottenere un nuovo indirizzo. Questa caratteristica è comodissima nella lettura o nell'aggiornamento di tabelle di dati, partendo da un indirizzo di base. Generalmente, IY viene usato dalle routine della ROM per puntare al centro dell'area riservata alle variabili di sistema. Questo registro, dopo l'uso, va riportato al valore originale altrimenti il computer si blocca.

Ci sono altri due registri speciali nel microprocessore Z8Ø: I e R. I si usa per contenere parte dell'indirizzo usato dalle *routine di interrupt*. Queste servono per interrompere, ogni cinquantesimo di secondo, il normale corso del programma, onde eseguire alcune funzioni essenziali (quali la scansione della tastiera). Il registro R, detto *di refresh*, non va usato né sullo Spectrum né sullo ZX81. RAMTOP coincide col fondo dello stack, che può occupare quanta RAM sia necessaria. Il puntatore dello stack, o registro SP, è a 16 bit per contenere l'intero indirizzo dell'ultimo byte occupato dallo stack.

Quando si usa il decimale codificato in binario?

Il BCD (Binary Coded Decimal) si usa per visualizzare numeri sullo schermo in modo rapido. Diversamente dall'hex, il BCD non va convertito in decimali per essere letto, essendo già in forma decimale.

Per codificare un numero decimale di due cifre in un byte binario, si mette la cifra a destra nei quattro bit meno significativi e la cifra a sinistra nei quattro bit più significativi (queste metà di byte sono talvolta denominate *nibble*).

In questo modo non si sfruttano tutte le combinazioni binarie offerte da un byte: su 16 cifre codificabili in quattro bit binari, se ne usano solo 1Ø. Il BCD è di fatto l'hex senza lettere.

L'unico problema sorge quando si ha un numero maggiore di 9 in uno dei *nibble*: ecco che interviene il flag di "mezzo riporto". (Negli Acorn il riporto è automatico).

L'Electron, il Micro BBC e il Vic 2Ø usano lo stesso chip, il 65Ø2; il Commodore 64 un chip molto simile, il 651Ø.

Ambedue hanno un solo accumulatore di otto bit e due registri indice di otto bit (X ed Y), che contengono un *offset* (scarto) da sommare a un indirizzo per averne un altro. Questo metodo di indirizzamento è comodo per consultare una tabella di dati in memoria. L'indirizzo di partenza, in questo caso, è quello che individua l'inizio della tabella, la cui lettura avviene usando un ciclo che incrementa l'offset ad ogni iterazione; cosicché viene letto, uno per volta, ciascun byte della tabella.

È possibile far sì che un programma in codice macchina legga un indirizzo contenuto in un altro indirizzo (che deve però essere nella pagina zero, poiché i contenuti di ogni locazione possono essere solo di otto bit). A entrambi gli indirizzi si possono aggiungere degli scarti (offset); va usato il registro X per lo scarto del primo indirizzo e Y per quello del secondo.

Il registro P, o *registro di processore*, chiamato sui Commodore *registro di condizione*, contiene i flag. Il flag di riporto (carry o C) corrisponde al bit Ø, il flag zero (Z) al bit 1, il flag V di overflow (traboccamento) al bit 6, il flag di segno (N) al bit 7.

Esistono anche altri due flag: il flag di decimale (D), nel bit 3, usato quando il microprocessore usa numeri in *decimale codificato in binario* (una rappresentazione binari dei normali numeri decimali). Nelle *routine di interrupt* si usano i flag di break e interrupt, (rispettivamente: B e I). Queste routine interrompono, a intervalli regolari, il corso normale dei programmi in codice macchina per eseguire funzioni essenziali, quali la scansione della tastiera. Lo stack si trova nella pagina uno, con la base in &Ø1FF; ne deriva che il puntatore allo stack (o registro SP) può essere di soli otto bit e contenere un qualsiasi valore compreso tra &ØØ e &FF, che rappresenta il byte meno significativo del primo indirizzo libero nello stack.

Il microprocessore sul Dragon e il Tandy ha due accumulatori, i registri A e B, utilizzabili come registri di otto bit o insieme come accumulatore D di 16 bit. Esistono due registri indice a 16 bit (X e Y), che contengono indirizzi ai quali è possibile aggiungere o togliere un valore, detto *offset* (scarto), per ottenere un nuvo indirizzo. Questo metodo è spesso usato per consultare tabelle di dati in memoria a partire da un indirizzo di base.

Il *registro del codice di condizione* contiene i flag. Il bit Ø corrisponde al flag riporto (C), il bit 1 al flag di overflow o traboccamento (V), il bit 2 al flag di zero (Z), il bit 3 al flag di segno (N), il bit 5 al flag di "mezzo riporto", che entra in gioco se si usano numeri in *decimale codificato in binario* (una rappresentazione binaria speciale dei numeri decimali). I bit 4, 6, e 7, corrispondenti al flag di maschera per le *interrupt normali* (I), al flag di maschera per la *interrupt veloce* (F), e al flag interno, E, si usano nelle *routine di interrupt*.

Queste interrompono, a intervalli regolari, il normale corso del programma in codice macchina per eseguire funzioni essenziali, quali l'aggiornamento del TIMER (contatempo). Il chip 6809, usato nel Dragon e nel Tandy, possiede anche un registro DP, di otto bit, chiamato *pagina diretta*, che contiene il byte più significativo degli indirizzi situati nella pagina diretta.

In questo chip esistono due puntatori allo stack, ambedue a 16 bit: S e U. Il puntatore S punta all'ultimo indirizzo pieno sullo stack di macchina (la cui base coincide con RAMPTOP, ossia &H7FFF, salvo che questo indirizzo non sia stato modificato per far posto a un programma in codice macchina). Il puntatore U serve invece per lo stack dell'utente, la cui base è di solito definita dal programmatore.

CAPIRE LE PEEK E LE POKE

Ecco come controllare direttamente il proprio computer, leggendo nella sua memoria e modificando o usando i valori in essa contenuti: e tutto ciò in BASIC

Il più delle volte si può usare un computer ignorando come funzioni internamente la sua memoria. Quando si scrive LET A = 67, per esempio, il computer si va a ricercare varie locazioni di memoria, tra quelle disponibili e le etichetta con 'A' e vi deposita il numero 67. Quando poi si scrive PRINT A, il computer sa esattamente dove andare a ritrovare il dato. Tutto ciò avviene automaticamente: soltanto in codice macchina si può e si deve dire al computer quali locazioni usare.

Ma anche in BASIC esiste un modo per esaminare la memoria del computer e usare nei programmi i valori in essa contenuti. Inoltre, si possono depositare nuovi valori, per modificare il comportamento del computer.

Gli strumenti offerti dal BASIC per questo socpo sono PEEK e POKE: PEEK consente di leggere il contenuto nella memoria, mentre con POKE si posono immettere i valori che si vogliono. I computer Acorn usano i simboli ? e ! al posto di PEEK e POKE, con risultati identici.

COME FUNZIONANO PEEK e POKE

Il Commodore, il Dragon, il Tandy, lo Spectrum 48K e gli Acorn hanno tutti 65536 locazioni di memoria indirizzabili, ossia 64K. I computer da 16K hanno 32768 locazioni. Una parte di questa memoria è a sola lettura (ROM) e i suoi contenuti sono fissi: possono venir letti con PEEK, ma non modificati con POKE. La restante memoria è ad accesso casuale (RAM), a completa disposizione del programmatore il quale vi può leggere e scrivere con la PEEK e le POKE e memorizzarvi i programmi in BASIC e le variabili. Questo programma permette di osservare ogni locazione di memoria ROM e RAM del computer:

```
10 INPUT "INDIRIZZO...";A
20 LET N=PEEK(A)
30 PRINT "CONTENUTO...";N
40 GOTO 10
```

```
10 INPUT "INDIRIZZO...",A
20 N=?A
30 PRINT "CONTENUTO..";N
40 GOTO 10
```

PEEK 12460

- CURIOSARE NELLA MEMORIA DEL COMPUTER
- USO DI **PEEK** E **POKE**
- **USO DI POKE** SULLO SCHERMO
- IL TIMER DELLO SPECTRUM
- DRAGON E TANDY A COLORI
- IL RECUPERO DI UN PROGRAMMA SUGLI ACORN
- IL COMMODORE SOTTO CONTROLLO

Si immetta qualsiasi numero tra Ø e 65535 e si vedrà cosa si trova in quella locazione, anche se i valori letti in certe aree di memoria possono non corrispondere all'effettivo contenuto.

Infatti, le locazioni nella RAM possono contenere valori che, mutando continuamente durante il funzionamento, sono difficili da "catturare".

Si noti come i valori siano sempre interi, compresi tra Ø e 255 (da Ø ad FF in hex), a conferma del fatto che sono tutti valori di un solo byte (un byte è un numero hex di due cifre). Un numero maggiore di FF non può entrare in un singolo byte: se si aggiunge 1 a FF in hex si ha Ø1ØØ, che occupa due byte e quindi due locazioni: Ø1 in una e ØØ nell'altra.

Il seguente programma permette di depositare numeri in memoria: per adesso si eseguano POKE con numeri di un solo byte, compresi cioè tra Ø e 255.

```
10 PRINT "CONTENUTO....";PEEK(30000)
20 INPUT "NUMERO......";N
30 POKE 30000,N
40 PRINT "NUOVO CONTENUTO";PEEK
   (30000)
45 PRINT
50 GOTO 20
```

(Per il Vic, si cambi la locazione 3ØØØØ in 9ØØ).

```
10 PRINT "CONTENUTO....";?3000
20 INPUT "NUMERO......",N
30 ?3000=N
40 PRINT "NUOVO
   CONTENUTO";?3000
45 PRINT
50 GOTO 20
```

Il programma, visualizzato il contenuto della locazione, vi deposita il numero voluto e ne visualizza nuovamente il contenuto a conferma dell'avvenuta modifica.

Si può cambiare a piacimento l'indirizzo sul quale operare (qui vale 3ØØØ). Si noterà che tutti i tentativi di modificare i valori delle ROM falliscono, senza arrecare alcun danno. Un maldestro impiego della POKE, in particolari aree della RAM, può tutt'al più provocare un blocco del sistema, ma non danni permanenti. Spegnendo e riaccendendo il computer, tutto torna come prima (salvo aver perso il programma!).

Si provi ora a depositare un numero superiore a 255. Il Dragon, il Tandy, il Commodore e lo Spectrum riporteranno un messaggio di errore, poiché ogni locazione può contenere un solo byte. Sui computer Acorn, invece, il numero viene accettato ma entra in memoria solo il byte più basso (il meno significativo). Pertanto, se il numero è 26Ø, cioè Ø1Ø4 in hex, si memorizza solo la parte Ø4.

USO DI POKE SULLO SCHERMO

Osservando le mappe di memoria alle pagine 2Ø8-215, si nota che una sezione di memoria è interamente dedicata all'immagine video.

In essa possono essere depositati certi valori, facendo comparire caratteri sullo schermo. Per visualizzare un particolare carattere sui computer Dragon, Tandy e BBC, occorre depositare il corrispondente codice ASCII, mentre sul Commodore l'appropriato codice video. Lo Spectrum adotta un metodo descritto più avanti. Si provino queste linee:

```
POKE 1024,1: POKE 55296,3
```

```
POKE 7680,1:POKE 38400,3
```

```
MODE 7: PRINT: ?&7C00=65
```

```
POKE 1024,65
```

Nell'angolo in alto a sinistra comparirà la lettera A.

Sul BBC e sul Dragon, lo schermo non deve essere fatto scorrere prima di depositare i caratteri che, altrimenti, possono apparire ovunque. Quindi, prima si digiti CLS per pulire lo schermo, poi RETURN o ENTER. Sul BBC si può preparare lo schermo con un MODE. Si noti che sull'Electron non si possono depositare caratteri con la stessa tecnica, in quanto esiste soltanto il modo teletext (MODE 7).

Sul Commodore sono necessarie due POKE, la prima per posizionare la A sullo schermo e la seconda per farla comparire in un colore diverso da quello dello sfondo. Ci sono dei vantaggi ad operare sullo schermo con POKE, anziché con le più familiari PRINT AT, PRINT TAB o PRINT @, anche se l'uso di queste ultime per posizionare i caratteri sullo schermo è più facile e comodo. Per esempio, visualizzando un carattere nell'ultima posizione dello schermo con una PRINT, l'intero schermo scorre in sù, cosa che non accade se si usa una POKE. Ciò è particolarmente utile nei giochi in cui si vuole spostare un carattere sfruttando l'intero schermo.

I CARATTERI DELLO SPECTRUM

Sullo Spectrum le cose sono diverse, dal momento che non si può usare una POKE per ottenere un intero carattere sullo schermo. I caratteri, come gli UDG, sono formati da punti disposti su una griglia 8 × 8. Occorre depositare singolarmente ciascuna linea sullo schermo, per riprodurre l'intero carattere. Per fortuna, le forme dei caratteri sono memorizzate nelle ROM, per cui si può andare a leggere ogni linea e poi depositarla sullo schermo. Il seguente programma visualizza una A sullo schermo in alto a sinistra:

```
10 LET dest=16384
20 FOR a=15880 TO 15887
30 POKE dest,PEEK a
40 LET dest=dest+256
50 NEXT a
```

La linea 10 assegna l'indirizzo dello schermo alla prima linea del carattere. Il ciclo FOR... NEXT scorre l'area della ROM che contiene la sequenza di valori per il carattere e la 30 fa comparire la linea sullo schermo. La linea 40 incrementa l'indirizzo dello schermo di 256 per posizione la sucessiva linea di punti immediatamente sotto alla precedente.

È un metodo piuttosto lento, per cui è quasi sempre preferibile, sullo Spectrum, usare PRINT AT o PRINT TAB.

LE PEEK IN ROM E IN RAM

Dal primo programma per osservare la memoria del computer non si ricavano che numeri, compresi tra 0 e 255. Molti di essi sono in realtà i codici ASCII di lettere che fanno parte di parole o di intere frasi. (Una tabella del codice ASCII è riprodotta a pagina 243).

Il seguente programma legge l'intera ROM del computer e converte i numeri in caratteri prima di visualizzarli sullo schermo:

```
10 FOR A=0 TO 16383
20 LET N=PEEK A
30 IF N>31 AND N<127 THEN
   PRINT CHR$N;
40 NEXT A
```

```
10 FOR A=40960 TO 49159
20 N=PEEK(A)
30 IF N>31 AND N<91 THEN
   PRINT CHR$(N);
40 NEXT A
```

Per il Vic, si cambi la linea 10 in:

```
10 FOR A=49152 TO 57347
```

```
10 FOR A=&8000 TO &FFFF
20 N=?A
30 IF N>31 AND N<127 THEN
   PRINT CHR$(N);
40 NEXT A
```

```
10 FOR A=&H8000 TO &HBFFF
20 N=PEEK(A)
30 IF N>31 AND N<127 THEN
   PRINT CHR$(N);
40 NEXT A
```

Le linee 10 e 40 passano in rassegna tutta la ROM, la 20 legge ogni locazione e la 30 converte il valore in un carattere per visualizzarlo. La linea 30 delimita anche l'arco di numeri da convertire perché il computer non tenti di visualizzare codici di controllo o simboli grafici.

Anche la RAM può essere esaminata allo stesso modo, semplicemente cambiando gli indirizzi di memoria alla linea 10. Gli indirizzi usati qui sotto puntano all'area in cui si immagazzinano i programmi BASIC. Si può così vedere come è in realtà memorizzato il programma stesso. Per esser sicuri che la memoria RAM sia sgombra dai rimasugli di altri programmi, è bene spegnere e riaccendere il computer. Ecco la nuova linea 10:

```
10 FOR A=23755 TO 65000
```

```
10 FOR A=16510 TO 30000
```

```
10 FOR A=2048 TO 40959
```

Si usi FOR A=4096 TO 7679 per i Vic espansi, ma si inizi da 1024 per quelli con soli 3K. Per gli altri, si parta da 4608.

```
10 FOR A=&H1E00 TO &H1F00
```

```
10 FOR A=PAGE TO PAGE+255
```

Si provi ad esaminare altre aree della RAM con lo stesso sistema.

L'USO DI PEEK E POKE

Abbiamo finora osservato la memoria del computer in generale e l'uso della POKE sullo schermo, facendoci un'idea del funzionamento di PEEK e POKE, ma non della loro utilità specifica.

Per far questo occorre considerare specifiche locazioni di memoria. Per esempio, sullo Spectrum, la locazione 23609 controlla il suono emesso dalla tastiera alla pressione dei tasti. Di norma essa contiene 0, che genera un breve clic, ma un numero maggiore, fino a 255, provoca un prolungato bip. Con POKE 23609,80 si ottiene un altro tipo di rumore.

Le possibilità della PEEK e della POKE dipendono dal computer che si possiede.

TABELLA DEI CODICI ASCII

Ogni carattere usato dal computer ha un numero di codice e la maggior parte dei computer adotta un sistema di codifica standard, chiamato ASCII (da American Standard Code for Information Interchange).

Tali codici sono quelli usati con CHR$ e ASC (o CODE sullo Spectrum). CHR$ converte il numero in un carattere, mentre ASC o CODE opera all'opposto, convertendo un carattere nel suo numero di codice.

Inoltre, quando il computer memorizza una parola è il codice ASCII dei caratteri a essere memorizzato.

Ecco la tabella dei codici ASCII:

Numero di codice	Carattere ASCII	Numero di codice	Carattere ASCII
32	spazio	62	>
33	!	63	?
34	"	64	@
35	#	65	A
36	$	66	B
37	%	67	C
38	&	68	D
39	'	69	E
40	(	70	F
41	)	71	G
42	★	72	H
43	+	73	I
44	,	74	J
45	–	75	K
46	.	76	L
47	/	77	M
48	0	78	N
49	1	79	O
50	2	80	P
51	3	81	Q
52	4	82	R
53	5	83	S
54	6	84	T
55	7	85	U
56	8	86	V
57	9	87	W
58	:	88	X
59	;	89	Y
60	<	90	Z
61	=		

Per esempio, le tastiere degli altri computer non emettono alcun suono, per cui non esiste una locazione di memoria dedicata a tale scopo. Oppure, ciò che su un computer richiede sei o sette PEEK e POKE, su un altro si ottiene con un solo comando BASIC.

Il Commodore ricorre più di tutti a PEEK e POKE. Lo Spectrum, il Dragon e il Tandy le usano occasionalmente, spesso per cambiare il funzionamento della tastiera o dell'immagine video. Sugli Acorn può capitare di non doverle mai usare in BASIC, mentre è possibile in linguaggio Assembly.

Ecco infine qualche prova da fare sul proprio computer.

Si è visto come cambiare il suono della tastiera. La prossima POKE modifica l'intervallo di attesa, prima che per un tasto si avvii l'autorepeat. Ciò torna utile, nei giochi, per sveltire il movimento di personaggi ottenuto tramite i tasti. In questo esempio e nei seguenti, X è una variabile da stabilire quando si digita la linea:

```
POKE 23561,X
```

Di norma, all'accensione del computer, X vale 35, perciò si usi un numero minore per diminuire l'intervallo e uno maggiore, fino a 255, per aumentarlo.

È anche posibile modificare lo stesso autorepeat usando:

```
POKE 23562,X
```

Questa volta X è in genere 5. Per un autorepeat veloce si usi X=1, per uno lento X=255.

TIMER

Lo Spectrum non possiede un comando TIME, quindi si ricorre alla PEEK per usare il timer interno al computer. Il tempo è contenuto in tre locazioni consecutive.

```
PRINT (PEEK 23672+256*PEEK 23673+
   65536*PEEK 23674)
```

Il programma ci dirà da quanti cinquantesimi di secondo è acceso lo Spectrum, dall'ultimo NEW. Il timer si azzera, infatti, con un NEW o depositando Ø in ciascuna delle tre locazioni.

ALTRE POKE

Ecco altre due POKE utili per la scrittura di programmi che altri poi useranno. La prima ci garantisce che le lettere digitate appaiano tutte in maiuscolo:

```
POKE 23658,8
```

Ciò è utile quando si desiderano immissioni omogenee. Per tornare al modo normale, si usi POKE 23658,Ø.

La seconda sostituisce al cursore un qualsiasi carattere o intere parole chiave, a seconda del valore della POKE.

```
10 FOR X=1 TO 255
20 POKE 23617,X
30 INPUT a$
40 NEXT X
```

La linea 3Ø serve soltanto per far sì che appaia un cursore sullo schermo. Appena digitato un qualsiasi carattere, la linea 2Ø si occupa di sostituire al cursore il simbolo successivo. I più interessanti si hanno per valori di X compresi tra 2ØØ e 23Ø: X=21Ø dà come cursore [CONTINUE] e X=225 dà un [?].

Un comune impiego di PEEK e POKE nell'attivare la ripetizione automatica dei tasti. Né il Dragon né il Tandy prevedono l'autorepeat, per cui muovere rapidamente oggetti sullo schermo risulta fastidioso e spesso comporta ripetute pressioni dei tasti. Adoperando delle PEEK per esaminare quale tasto viene premuto, si riesce ad ottenere un effetto identico all'autorepeat.

Un simile metodo è già stato impiegato nella sezione "Giochi al Computer". Alla pressione di un tasto, viene depositato uno speciale codice in una tra 6 locazioni di memoria. La PEEK le esamina per vedere quale tasto viene premuto. È un metodo veramente facile: il seguente programma, ad esempio, usa i tasti cursore per disegnare sullo schermo:

```
10 PMODE Ø,1
20 PCLS
30 SCREEN 1,1
40 X=127:Y=95
50 IF PEEK(341)=223 THEN Y=Y-2
60 IF PEEK(342)=223 THEN Y=Y+2
70 IF PEEK(343)=223 THEN X=X-2
80 IF PEEK(344)=223 THEN X=X+2
90 IF X<Ø OR X>255 THEN X=-255*(X
   <Ø)
100 IF Y<Ø OR Y>191 THEN Y=-191*
   (Y<Ø)
110 PSET(X,Y,5)
120 GOTO 50
```

Sul Tandy i sostituisca 233 con 247, alle linee da 5Ø a 8Ø.

Le tabelle alle pagine 245 e 246 mostrano quale codice sia associato ad ogni tasto e la locazione corrispondente.

Premendo il tasto [A], per esempio, PEEK (239) fornisce il codice 251 sul Dragon e 254 sul Tandy. Adesso si capirà da dove provengono i valori usati per le PEEK nell'ultimo programma.

SCHERMO MULTICOLORE

Il programma che segue visualizza linee di diverso colore sullo schermo che, accostate l'una all'altra, producono interessanti sfumature. Premendo qualsiasi tasto si può fermare il formarsi dell'immagine ed esaminare il valore corrispondente alla particolare

```
10 PMODE 3,1:SCREEN 1,Ø:PCLS3
20 FOR K=Ø TO 255
```

```
30 POKE 178,K
40 LINE(78,46)—(178,146),PSET,BF
50 IF INKEY$<>"" THEN 80
60 NEXT
70 GOTO 70
80 CLS:PRINT K
```

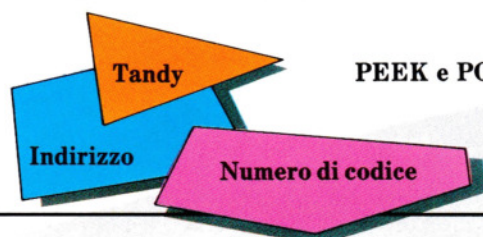
sfumatura. Il colore del primo piano è regolato dal contenuto della locazione 178, che normalmente varia da Ø a 3, offrendo quattro colori, ma se ne ottengono altri striati depositando valori superiori a 3. Alla linea 4Ø si usa PSET per i colori del primo piano. Per quelli dello sfondo si sostituisca PSET con PRESET e, alla linea 3Ø, 178 con 179. Bloccando il programma, possiamo esaminare (ed eventualmente annotare per usi futuri) il valore di K per una particolare sfumatura di colore.

PEEK e POKE relative alla tastiera

Dragon — Indirizzo — Numero di codice

	191	223	239	247	251	253	254
338	ENTER	X	P	H	@	8	Ø
339	CLEAR	Y	Q	I	A	9	1
340		Z	R	J	B	:	2
341		↑	S	K	C	;	3
342		↓	T	L	D	,	4
343		←	U	M	E	—	5
344		→	V	N	F	.	6
345		SPACE	W	O	G	/	7

CAMBIO DI VELOCITA'

Ecco infine un modo per velocizzare il computer, utile per accelerare un gioco avendo superato un certo punteggio. Si usi la prima POKE per accelerare, la seconda per tornare

PEEK e POKE relative alla tastiera

	191	223	239	247	251	253	254
338	ENTER	8	Ø	X	P	H	@
339	CLEAR	9	1	Y	Q	I	A
340		:	2	Z	R	J	B
341		;	3	↑	S	K	C
342		,	4	↓	T	L	D
343		—	5	←	U	M	E
344		.	6	→	V	N	F
345		/	7	SPACE	W	O	G

alla norma:

```
POKE 65495,1
POKE 65494,1
```

(Qualsiasi numero, al posto di 1, produce lo stesso effetto.) Il problema con questa POKE è che non funziona su tutti i microprocessori, ma val la pena di provarla sul proprio apparecchio.

Per chi pensa che i giochi vadano troppo veloci, ecco come rallentare l'emissione su schermo:

```
POKE 359,6Ø
```

Con POKE 359,57 si torna alla norma. Non si usi questo artificio sul Tandy, perché disabilita il ritorno automatico allo schermo testuale.

PEEK e POKE sono due comandi grazie ai quali si riesce a tenere sotto totale controllo il Commodore. Senza di essi ci sarebbero ben poche possibilità, in BASIC, di arricchire le prestazioni sonore e grafiche della macchina. Per sperimentare le POKE usate più di rado, è indispensabile procurarsi un'accurata mappa della memoria, in cui appaia la funzione di ciascuna locazione (spesso simili mappe sono riportate in appendice ai manuali).

È bene sottolineare che non deriva alcun danno all'apparecchio, usando delle POKE anche nelle zone cosiddette *riservate*. Ispezionando determinate aree, invece, si otterranno sequenze di numeri senza senso.

Nella peggiore ipotesi, del resto abbastanza rara, un'incauta POKE può al massimo bloccare il computer e occorrerà spegnerlo e riaccenderlo. Seguono alcuni esempi che danno un'idea sul controllo possibile mediante le PEEK e le POKE.

CAMBIAMENTI ESTETICI

Due delle POKE più usate sul Commodore 64 riguardano la definizione dei colori dello schermo e della cornice:

POKE 5328Ø,X per la cornice
POKE 53281,X per lo schermo

Il valore X può andare da Ø a 15 in corrispondenza ai codici dei colori elencati nel manuale. Si provi a depositare in ambedue le locazioni il valore Ø: si otterrà uno schermo nero con cursore blu. Per cambiare basta premere CTRL e uno qualsiasi dei tasti colore corrispondente alla combinazione desiderata. Una combinazione di bianco (o giallo) su nero può riuscire di maggior gradimento che non le normali combinazioni di blu e meglio si adatta al word processing o alla gestione di archivi dati che non a soluzioni più colorate. Altre POKE in questa zona di memoria (53248-54271) si riferiscono all'uso dei vari modi grafici, compresi gli sprite, perché questa parte della memoria concerne il chip VIC.

CONTROLLO DEI TASTI

Nei giochi oppure nel disegno questa è una POKE di frequentissimo impiego:

```
POKE 65Ø,X
```

ogni valore di X maggiore di 128 abilita la ripetizione automatica dei tasti e un successivo valore minore riporta le cose allo stato normale. Quanto alla protezione dei propri programmi, non esiste un metodo sicuro, ma con una o due POKE si può complicare la vita agli intrusi:

```
POKE 775, 2ØØ
```

Questa rende impossibile usare LIST per esaminare un programma: si ristabilisce la situazione normale tornando a depositare 167. Si può impedire l'uso del tasto RUN/STOP con:

POKE 8Ø8, 239 (POKE 8Ø8, 237)

Si noti che il secondo POKE in programma annulla gli effetti del primo e che i restanti POKE si presentano pure in questo formato. Il secondo POKE deve ovviamente essere digitato (senza numero di linea) prima che possano essere riprese le normali funzioni della tastiera. Questo POKE può risultare utile nei sottoprogrammi che prevedono immissioni da tastiera. RUN/STOP e RESTORE si possono anche disabilitare:

POKE 8Ø8, 251 (POKE 8Ø8, 237)

È interessante notare come la stessa locazione si usi per entrambi i metodi di protezione. Ciò accade di rado e suggerisce un importante uso di POKE, quando si tratta di locazioni con funzioni multiple: la commutazione dei bit di ogni byte in alto (on) o in basso (off).

Il valore massimo 255, per esempio, si ottiene se tutti i bit di un byte sono 'alti': osserviamo i valori dei singoli bit:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bitØ
128	64	32	16	8	4	2	1

Il risultato della somma di tutti i valori di questi bit vale 255. Di tutti gli altri valori usati con POKE 8Ø8, il 237 si ottiene solo quando il bit 4 e il bit 1 sono bassi, ossia *off* (cioè 255 meno 16, meno 2). Il valore 234, che riporta il computer al funzionamento normale, si otiene quando i bit 4,2 e 1 sono bassi. Il valore 239 di POKE 8Ø8 pone il bit 1 in condizione *on*, disabilitando il tasto RUN/STOP.

MANIPOLAZIONE DELLA MEMORIA

Spesso le PEEK e le POKE servono per manipolare i confini della memoria, in modo da proteggere varie aree di RAM da sovrapposiozioni da parte del BASIC. Inoltre, si possono assegnare diverse locazioni di partenza alle locazioni dello schermo, oppure si possono temporaneamente ussare locazioni normalmente adibite a funzioni diverse (come il buffer I/O del registratore). Si veda alle pagg. 214-215.

Le locazioni modificabili tramite POKE si possono considerare come 'celle' per l'immagazzinamento di informazioni, il cui valore va da Ø a 255. Tali informazioni possono agire da commutatori, per il colore dello schermo o un particolare suono.

Ciascuna cella ha un unico e preciso compito. I valori sono per lo più predefiniti o fissi, ispezionabili con il programma descritto all'inizio di questa lezione.

Spesso è utile ripulire queste locazioni da informazioni vecchie o inutili, tramite una semplice POKE con valore Ø. È interessante provare a depositare un valore in un'area di sovrapposizione ROM/RAM (sul Commodore 64 questa corrisponde alle locazioni dal decimale 4Ø96Ø in sù).

Leggendo con una PEEK, si ottengono i valori concernenti la ROM del BASIC, ma usando la POKE si depositano valori nell'area RAM 'sottostante'. Ecco quindi perché nei listati per il Commodore 64 appare spesso una linea del tipo POKE N, PEEK (N). Il seguente è un programma per copiare la ROM del BASIC nella sottostante RAM:

```
10 FOR N=40960 TO 49151
20 POKE N, PEEK (N) : NEXT
```

Si esegua e si attenda circa un minuto, finché non compare il cursore: il BASIC è ora in RAM e quindi in una posizione dove può venir modificato. Ma prima occorre "liberarsi" della ROM. Si digiti:

```
POKE 1,54
```

Stiamo agendo sulla locazione forse più importante di tutta la memoria. Come risultato, si ottiene una versione di BASIC "vulnerabile" e l'uso di POKE, in quest'area di memoria (4Ø96Ø-49151), produce interessanti conseguenze. In modo diretto si digiti questa stringa di PEEK.

```
PRINT PEEK(41229),PEEK(41230),PEEK
    (41231),PEEK(41232)
```

Premendo RETURN, si ottengono quattro numeri: 76, 73, 83 e 212. Dalla tabella ASCII si vede che questi valori corrispondono ai codici per L, I, S e T: abbiamo individuato la parola chiave LIST!

Supponiamo di voler modificare il nome del comando: basta depositare nelle stesse locazioni di memoria gli opportuni codici ASCII (sommando 128 al quarto codice). Sostituiamo LIST con SHOW:

```
POKE 41229,83: POKE41230,72
    : POKE 41231,79: POKE 41232,215
```

Premendo RETURN, si ottiene il listato del precedente programma. Per ripristinare le condizioni normali, si premano simultaneamente RUN/STOP e RESTORE (o si usi POKE 1,55 se si vuole ottenere lo stesso scopo *via software*). Il procedimento si applica anche a tutti gli altri comandi.

Gli Acorn non possiedono le funzioni PEEK e POKE, ma degli *operatori indiretti*: caratterizzati dai simboli ?, ! e l'operatore stringa $. Il punto interrogativo si è già visto all'opera: gli altri sono spiegati nel riquadro in fondo a questa lezione. L'Acorn, comunque, raccomanda di evitare l'uso di questi operatori nei programmi BASIC, dato che comprometterebbero il funzionamento di programmi trasferiti, tramite il sistema a rete *Tube*, ad altri microprocessori con diverse mappe di memoria. Virtualmente, tutte le funzioni che si possano desiderare da un programma BASIC sono ottenibili coi i comuni comandi.

Occasionalmente, però, può capitare di dover intervenire direttamente sulla memoria del computer con ?, ! o $. Un classico esempio è quello in cui, in seguito a una involontaria pressione di BREAK, si abbia poi scritto ØLD (oppure Ø.), anziché OLD (o O.). Chiedendo un LIST, si ottiene una linea Ø, seguita da LD o da un puntino. Il computer ha creduto che stessimo immettendo un nuovo programma e il vecchio sembra scomparso. Per fortuna, si può recuperarlo. Scrivendo ØLD o Ø., abbiamo alterato i primi byte del progrmma, alterando ciò che vi si trovava, ma non il resto. Si digiti questa linea, ma prestando attenzione a non farla precedere da un numero di linea, per non peggiorare le cose:

```
FOR A%=7 TO 260: IF?(PAGE+A%)=
    &0D THEN !(PAGE+4)=&2020F420:?
    (PAGE+3)=A% ELSE NEXT
```

PAGE è l'inizio del programma BASIC, A% è un contatore e &ØD è il codice del ritorno carrello (il computer lo accoda a ogni linea di programma). Il programma legge la memoria a partire da PAGE+7, ignorando così i primi sette byte danneggiati. Quando incontra il codice di "ritorno carrello", che indica l'inizio di una nuova linea, viene depositata in memoria una serie di 4 byte a partire da PAGE+4. I 4 byte sono &2020F420 e rappresentano una REM (codice F4), preceduta da due spazi e seguita da un altro spazio. Si inizia da PAGE+4, per lasciare intatto il numero di linea Ø. Al termine, ? (PAGE+3)=A% deposita la corretta lunghezza della linea al punto giusto. Un LIST del programma darà ora:

```
0 REM
```

seguito dal resto del programma, mancante, al massimo, delle prime due linee.

Dopo un programma utile e pratico ecco un paio di esempi futili, per il BBC. Entrambi sono destinati ai dispositivi interni di controllo. Nell'area in questione non si dovrebbe andar depositando valori senza sapere esattamente cosa si sta facendo, ma questo esperimento non comporta alcun rischio (in caso di problemi, si prema BREAK o si spenga per un momento il computer). La seguente linea trasforma la tastiera in una specie di strano strumento musicale:

```
?&FE40=0
```

Questa, invece, altera il timer in modo da rallentare le operazioni del computer:

```
?&FE45=1 : ?&FE46=0
```

Per ripristinare la normalità si prema BREAK.

GLI OPERATORI INDIRETTI DEGLI ACORN

Sui computer Acorn il simbolo ? (detto *operatore indiretto*) si usa al posto di PEEK e POKE. Un secondo operatore indiretto è !, che dà il contenuto di quattro locazioni di memoria consecutive e permette di usare PEEK e POKE relativamente a quattro byte per volta. L'utilità è evidente, quando occorre memorizzare numeri grandi. Per esempio:

```
!3000=99999
```

memorizza il valore in quattro byte, a partire dalla locazione 3ØØØ. Per visualizzare il contenuto di quattro locazioni alla volta si usi:

```
PRINT !3000
```

Il punto esclamativo è utile quando si ha a che fare con variabili intere (tutte quelle con un segno di percentuale in fondo, come anno%), che vengono memorizzate in quattro byte. Il terzo e ultimo operatore indiretto, $, è quello relativo alle stringhe e permette la sistemazione di una stringa in memoria. I codici ASCII dei vari caratteri che compongono la sequenza vengono depositati in altrettante locazioni di memoria consecutive, accodando il codice di ritorno carrello all'ultimo carattere. Si provi questo:

```
10 A=3000
20 $A="WORD"
30 PRINT  $A: PRINT
40 FOR I=0TO 4
50 PRINT?(A+I);"□";CHR$?(A+I)
60 NEXT I
```

La linea 2Ø deposita la stringa in memoria e la 3Ø visualizza tutti i caratteri partendo dalla locazione A fino al codice di ritorno carrello. Le altre linee servono per dimostrare che la stringa è davvero memorizzata nel modo descritto. Il codice ASCII e il carattere corrispondente compaiono uno alla volta, mentre il programma scorre la memoria dalla locazione 3ØØØ alla 3ØØ4.

DRAGON E TANDY: UNA MIGLIORE GRAFICA

Dragon e Tandy hanno cinque modi grafici, o PMODE, ad alta risoluzione numerati da Ø a 4. Essi differiscono nel grado di risoluzione, ossia nella precisione dell'immagine su schermo, e nei colori: i PMODE prevedono l'impiego di due oppure di quattro colori, e ognuno di essi offre una scelta sul *set di colore*.

Quando si usano i modi a quattro colori, PMODE 1 e 3, si osserva che il set di colore Ø consiste di Verde, Giallo, Blu e Rosso, mente il set di colore 1 consiste di Nocciola (Bianco), Azzurro, Magenta e Arancio. Il comando SCREEN esegue la selezione del set di colore, SCREEN 1,Ø per il set Ø e SCREEN 1,1 per il set 1.

La differenza tra PMODE 1 e PMODE 3 consiste nella risoluzione: la grafica prodotta nel PMODE 1 è più grossolana, perché di minor risoluzione che non quella prodotta nel PMODE 3.

I due pregi del PMODE 1 sono quello di usare meno memoria e quello di una grafica più rapida.

Nell'impiego del PMODE 4 per disegnare UDG in bianco e nero, ogni pixel può essere individualmente *acceso* o *spento* usando i valori binari 1 e Ø. Invece, nel PMODE 3, i pixel funzionano a blocchi di due e nel PMODE 1 a blocchi di quattro. Presto si vedrà come i numeri binari di 2 bit controllino il colore di questi blocchi di pixel. La tabella 1 offre una lista di colori e delle risoluzioni disponibili in ogni PMODE.

LA DEFINIZIONE DI UDG A COLORI

Si digiti questo programma e lo si esegua:

```
10 PMODE3,1:PCLS:SCREEN1,0
20 FOR I=1 TO 9
30 READ A,B:POKE 1800+I*32,A:POKE 1801
   +I*32,B
40 NEXT
50 FOR K=10 TO 22:POKE 1800+K*32,192
   :POKE 1801+K*32,0:NEXT
60 GOTO 60
70 DATA 192,0,213,149,213,149,213,149,
   234,170,234,170,213,149,213,149,213,
   149
```

Abbiamo già visto come definire UDG monocromi (pagine 38-41). Aggiungiamo ora la colorazione alla grafica di Dragon e Tandy con alcune tecniche adatte a questi apparecchi

- I DIVERSI SET DI COLORE E IL LORO USO
- LA DEFINIZIONE DI UDG A COLORI
- ALZIAMO UNA BANDIERA
- ESPLORIAMO I **PMODE**

Il programma disegna una bandiera con l'asta (secondo un metodo simile a quello di pagina 4Ø per produrre UDG in bianco e nero), depositando elementi di DATA sullo schermo, mediante delle POKE. Stavolta, per osservare l'UDG più chiaramente, le DATA vengono depositate in memoria a partire dalla locazione 18ØØ (nella parte centrale dello schermo), anziché dalla 1536 (in alto a sinistra).

La linea 1Ø seleziona il PMODE 3, il set di colore Ø (Verde, Giallo, Blu e Rosso) e ripulisce lo schermo per l'UDG. Le linee da 2Ø a 4Ø depositano nove coppie di DATA relative alla bandiera. Il resto del carattere è disegnato ripetendo più volte i codici 192 e Ø.

La linea 5Ø disegna l'asta ripetendo 22 volte la POKE di 192 e Ø, in modo più comodo di quello usato in precedenza, perché evita di dover digitare altri 44 elementi di DATA alla linea 7Ø.

La linea 6Ø provvede, nel modo consueto, a mantenere abilitato lo schermo ad alta risoluzione, per poter osservare l'UDG. La differenza principale col programma originario è nella frase DATA: i valori sono in decimale ed è stata convertita partendo da numeri binari di 8 bit con lo stesso metodo dell'UDG in bianco e nero di pagina 38. Stavolta, però, invece di *singoli* bit che comunicano al computer di accendere o spegnere *singoli* pixel, sono *coppie* di bit che comunicano al computer quale colore attribuire a *coppie* di pixel. Nel PMODE 1 un paio di bit definisce due coppie di pixel, una sotto l'altra.

La **figura 1** mostra come la bandiera è realizzata con coppie di pixel: si possono scrivere coppie di valori binari in ciascuno dei blocchi di due pixel; il colore selezionato dipende dal set di colore scelto. Nel set di colore Ø, si scrive ØØ sul blocco verde, Ø1 sul giallo, 1Ø sul blu e 11 sul rosso. Quattro di questi numeri di due bit costituiscono ciascun elemento di 8 bit della DATA. Se si è scelto il set di colore 1, digitando SCREEN 1,1 alla linea 1Ø, allora la bandiera risulta Nocciola, Azzurro, Magenta e Arancio. In questo set si scrive ØØ per un blocco nociola, Ø1 per uno azzurro, 1Ø per uno magenta e 11 per uno arancio.

Si cambi alla linea 1Ø il PMODE 3 in PMODE 1, si lanci il programma e comparirà sullo schermo una versione allungata della bandiera. Ricordiamo che nel PMODE 1 ogni elemento della DATA è ora relativo a quattro pixel e non a due. Volendo definire un UDG nel PMODE 1, occorre lavorare su blocchi di 2×2 pixel e si scriveranno numeri di 2 bit a rappresentare il colore sulla linea in cima a ciascun blocco.

192	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
213	1	1	Ø	1	Ø	1	Ø	1	1	Ø	Ø	1	Ø	1	Ø	1	149
213	1	1	Ø	1	Ø	1	Ø	1	1	Ø	Ø	1	Ø	1	Ø	1	149
213	1	1	Ø	1	Ø	1	Ø	1	1	Ø	Ø	1	Ø	1	Ø	1	149
234	1	1	1	Ø	1	Ø	1	Ø	1	Ø	1	Ø	1	Ø	1	Ø	17Ø
234	1	1	1	Ø	1	Ø	1	Ø	1	Ø	1	Ø	1	Ø	1	Ø	17Ø
213	1	1	Ø	1	Ø	1	Ø	1	1	Ø	Ø	1	Ø	1	Ø	1	149
213	1	1	Ø	1	Ø	1	Ø	1	1	Ø	Ø	1	Ø	1	Ø	1	149
213	1	1	Ø	1	Ø	1	Ø	1	1	Ø	Ø	1	Ø	1	Ø	1	149
192	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
192	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
	1	1															
	1	1															
	1	1															
	1	1															

SIMULAZIONE DEL MODO GRAFICO

Ecco un programma per osservare l'effetto delle DATA decimali sui pixel e sul loro colore in ognuno dei PMODE.

```
10 CLS
20 INPUT "MODO GRAFICO (0—4)□";MO
30 MO=INT(MO):IF MO<0 OR MO>4
   THEN 20
40 INPUT "NUMERO DELLO SCHERMO (0—
   1)";ST
50 ST=INT(ST):IF ST<0 OR ST>1
   THEN 40
60 INPUT "NUMBER (0—255)□";NU
70 NU=INT(NU):IF NU<0 OR NU>255
   THEN 60
80 IF MO<4 THEN LE=2 ELSE LE=1
90 IF MO<2 THEN DE=2 ELSE DE=1
100 IF (MO AND 1)=1 THEN SP=—
   2 ELSESP=—1
110 FOR K=7 TO 0 STEP SP
120 FOR L=1 TO LE
130 FOR J=1 TO DE
140 IF SP =—1 THEN PP = 1358+32*J+
   (3—K)*LE+L:CO = INT(.5+(NU AND 2
   ∧K)/2∧K):GOTO 160
150 PP=1393+32*J+L—K:CO=INT(.5+
   (NU AND 3*2∧(K—1))/2∧(K—1))
160 POKE PP,113—SP*15+CO*(14—SP)+
   ST*64
170 NEXT J,L
180 POKE PP—32*DE,47+K
190 POKE PP—32*DE+1+SP,48+K
200 IF SP=—2 THEN POKE PP+31,112—
   (CO>1)
210 POKE PP+32,112+(CO AND 1)
220 NEXT K
230 PRINT@482,"PREMERE QUALSIASI
   TASTO PER CONTINUARE";
240 SCREEN 0,1—ST
250 A$ = INKEY$:IF A$ = "" THEN 250
260 GOTO 10
```

Quando si impartisce un RUN, all'operatore viene chiesto di selezionare un modo grafico e un numero di schermo. In seguito, si digiti un numero decimale tra Ø e 255. Compariranno sullo schermo i colori e la relativa dimensione dei blocchi pixel, con sopra il corrispondente numero di bit e sotto gli equivalenti binari. Si usi questo programma per esplorare le varie possibilità. In una prossima lezione si imparerà ad animare i propri UDG.

NUMERO DI PMODE	PIXEL	SET DI COLORI DISPONIBILI	
		SCREEN 1,Ø	SCREEN 1,1
Ø	■■ ■■	Nero, Verde	Nero, Nocciola
1	■■ ■■	Verde, Giallo, Blu, Rosso,	Nocciola, Azzurro, Magenta, Arancio
2	■■	Nero, Verde	Nero, Nocciola
3	■■	Verde, Giallo, Blu, Rosso	Nocciola, Azzurro, Magenta, Arancio
4	■	Nero, Verde	Nero, Nocciola

LA MATEMATICA PER CREARE IMMAGINI

Mettiamo in azione alcune delle meno usate funzioni matematiche e vediamo come adoperare SIN, COS e TAN per disegnare curve, cerchi, ellissi e immagini basate su queste figure

I computer e la matematica formano una coppia inscindibile e, benché per la programmazone BASIC non occorrano particolari abilità in materia, molti dei procedimenti del BASIC sono familiari a ogni matematico. Tuttavia, molte funzioni del BASIC non sono impiegate solo nei calcoli, ma ricorrono in ogni genere di operazione. Tra le funzioni matematiche più utili ci sono quelle che servono a calcolare il rapporto tra angoli e distanze, che trovano applicazioni anche in campi apparentemente lontani dalla matematica, quale quello della grafica.

Supponiamo adesso di voler disegnare un orologio.

Disponendo del comando CIRCLE, si può facilmente disegnare il contorno del quadrante, ma poi non è affatto semplice collocare i numeri al posto giusto con delle PRINT, calcolando le coordinate una per una a mano. E un orologio con numeri fuori posto serve a ben poco!

Per fortuna, tale ingrato compito può essere assegnato alle funzioni matematiche del computer. Per tornare all'esempio dell'orologio, i numeri occupano ognuno una posizione, corrispondente a un dodicesimo del percorso intorno al cerchio, che il computer può calcolare fornendogli un numero in gradi o radianti, ambedue misure di grandezza di un angolo.

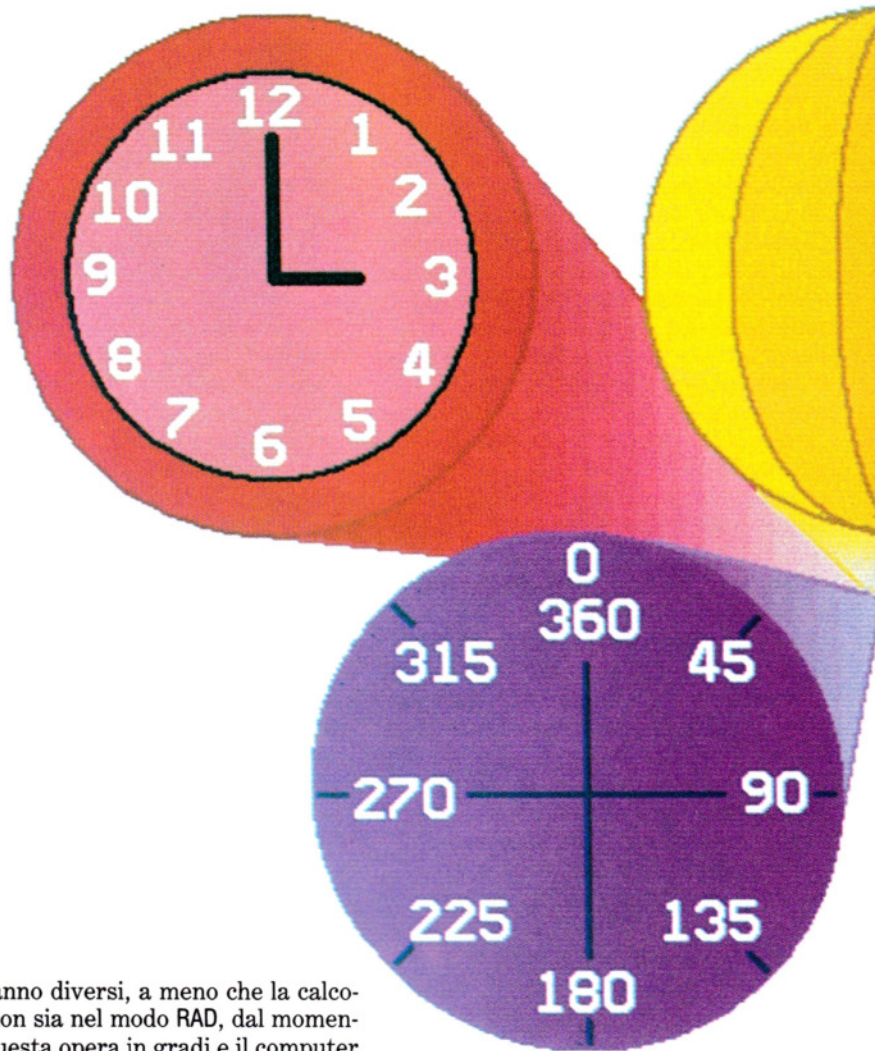
In un cerchio ci sono 360 gradi o 2 volte pigreco (PI) radianti. Sul quadrante dell'orologio ci sono 30 gradi (30°), o pigreco diviso 6, tra un numero e l'altro, ossia un dodicesimo di cerchio. Pigreco è un numero che ricorre spesso nella misurazione dei cerchi. Ad esempio, nel calcolo dell'area o della circonferenza. Pigreco (spesso indicato con la lettera π) vale circa 3,14 e molti computer (ma non il Dragon e il Tandy) conservano tale valore in memoria. Se si è perplessi sulla necessità di conoscere sia gradi che radianti o se si è propensi ad accettare uno solo dei due sistemi, riflettiamo sul fatto che gli uomini misurano comunemente in gradi, mentre i computer in radianti: per evitare confusione è bene, quindi, conoscerli entrambi.

Se si possiede una calcolatrice tascabile, si calcoli il seno di 30, poi lo stesso calcolo lo si faccia sul computer. I due risultati saranno diversi, a meno che la calcolatrice non sia nel modo RAD, dal momento che questa opera in gradi e il computer in radianti.

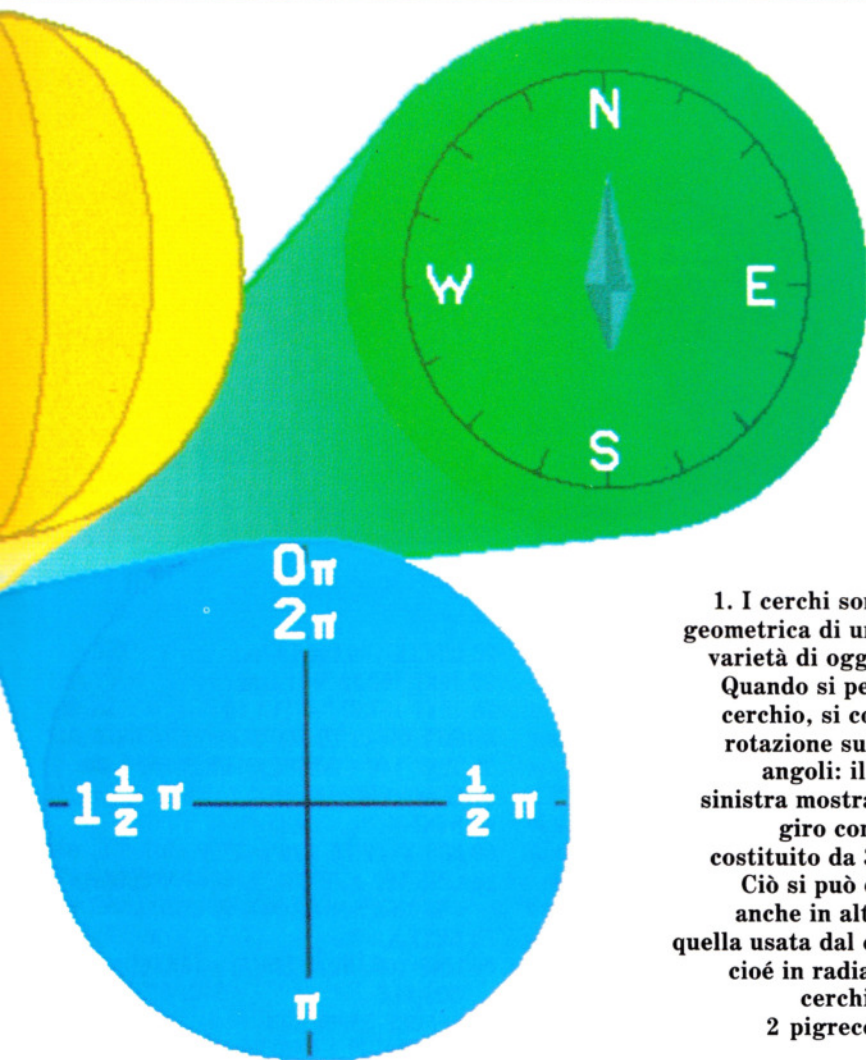
DA GRADI A RADIANTI

La conversione tra radianti e gradi è molto semplice: per ottenere un numero in radianti da un numero in gradi, si divide il numero per 180 e si moltiplica il risultato per pigreco. Il risultato inverso si ha moltiplicando il numero in radianti per 180 e poi dividendo per pigreco.

Il seguente programma è utile per esercitarsi in queste conversioni.

Si provi anche a confrontare i risultati con quelli forniti da una calcolatrice tascabile, dotata delle funzioni trigonometriche SIN, COS e TAN.

Non viene data una versione per gli Acorn, poiché questi possiedono due funzioni (RAD e DEG) che servono proprio a questo scopo.

```
10 INPUT "CONVERSIONE DA GRADI A
   RADIANTI(1) O DA RADIANTI A
   GRADI(2)?";a
20 IF a = 2 THEN GOTO 80
```

- CONVERSIONE DA GRADI A RADIANTI
- DISEGNO DI UNA BUSSOLA
- MISURARE GLI ANGOLI SU UNA CIRCONFERENZA
- SIGNIFICATO DI **SIN, COS** E **TAN**
- ANDAMENTO GRAFICO DELLE FUNZIONI **SIN** E **COS**
- USARE **SIN** E **COS** PER DISEGNARE CERCHI
- COSTRUIRE UNA SFERA PARTENDO DA ELLISSI

1. I cerchi sono la base geometrica di una grande varietà di oggetti reali. Quando si percorre un cerchio, si compie una rotazione suddivisa in angoli: il cerchio a sinistra mostra come un giro completo sia costituito da 360 gradi. Ciò si può esprimere anche in altra forma, quella usata dal computer, cioé in radianti: in un cerchio ci sono 2 pigreco radianti

```
30 IF a< >1 THEN GOTO 10
40 INPUT "NUMERO DA CONVERTIRE?";b
50 PRINT "IL VALORE IN RADIANTI
   È□";b/180*PI;
60 PRINT "QUALSIASI TASTO PER
   CONTINUARE": PAUSE 0: CLS : GOTO 10
70 INPUT "NUMERO DA CONVERTIRE?";b
80 PRINT "IL VALORE IN GRADI
   È□";b*180/PI;
90 PRINT "QUALSIASI TASTO PER
   CONTINUARE": PAUSE 0: CLS : GOTO 10
```

```
10 PRINT "[][π] CONVERSIONE DA GRADI
   IN"
20 PRINT "RADIANTI(1) O DA RADIANTI IN
   GRADI(2)":INPUT A
30 IS A<1 OR A>2 THEN 10
40 INPUT "[][]NUMERO DA CONVERTIRE";
   B:PRINT "[][][]VALE";
50 IF A=1 THEN PRINT
   B/180*π;"RADIANTI"
60 IF A=2 THEN PRINT B*180/π; "GRADI"
70 PRINT "[][][]QUALSIASI TASTO PER
   CONTINUARE"
80 POKE 198,0:WAIT 198,1:RUN
```

```
10 PI=4*ATN(1)
20 CLS
30 INPUT"CONVERSIONE DA GRADI IN
   RADIANTI(1) OPPURE DA RADIANTI IN
   GRADI(2)";A
40 IF A=2 THEN GOTO 90
50 IF A< >1 THEN GOTO 20
60 INPUT"NUMERO DA CONVERTIRE□";B
70 PRINT"VALE□";B*PI/180;"□RADIANTI"
80 GOTO 110
90 INPUT "NUMERO DA CONVERTIRE□";B
100 PRINT "VALE□";B*180/PI;"□GRADI"
110 PRINT "QUALSIASI TASTO PER
   CONTINUARE"
120 A$=INKEY$:IFA$=" " THEN GOTO 120
130 GOTO 20
```

Quando si esegue il programma, viene chiesto quale tipo di conversione operare: si risponda 1 o 2. Poi viene chiesta l'immissione del numero da convertire e, quasi istantaneamente, compare la risposta.

Fin qui tutto bene, ma è difficile immaginarsi un angolo rappresentato soltanto da un numero: sarebbe più facile se vedessimo l'angolo disegnato sullo schermo.

Una familiare rappresentazione di tutti gli angoli possibili è quella offerta dal goniometro, sul cui arco sono indicati, come in un righello, i vari angoli. Si potrebbe tracciare quindi un disegno sulla carta, ma perché confondersi, se abbiamo un computer che è in grado di produrre ottimi disegni?

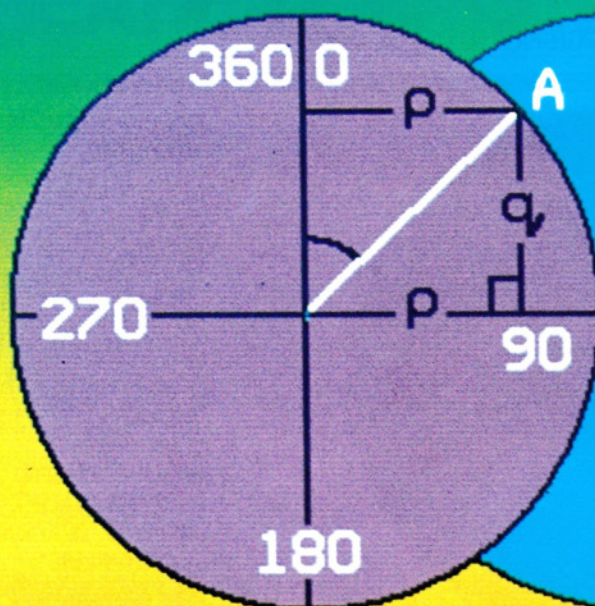
DISEGNARE UN CERCHIO

Basandosi sul nesso esistente tra il cerchio e la misurazione di un angolo, si può disegnare qualsiasi oggetto di forma circolare: da un semplice tondo al quadrante di una bussola. Il programma che segue disegna la superficie di un cerchio e vi colloca i numeri che segnano i gradi.

Nord è a 0, Est a 90, Sud a 180 e Ovest a 270 gradi.

Dopodiché potremo visualizzare a piacimento qualsiasi angolo.

Nella versione Dragon/Tandy, attorno al cerchio non appaiono i numeri, poiché, come si sa, il PMODE 4 (usato per disegna-

2. Se A si muove in senso orario, p diventa più grande e q più piccolo

re il cerchio) non consente di visualizzare contemporaneamente grafica e testo.

Più avanti nella lezione vedremo come ovviare.

Quando si esegue un RUN, il computer traccia una linea che va dal centro del cerchio a un punto sulla circonferenza, formando così un angolo della grandezza richiesta. Immettendo 90, quindi, la linea parte dal centro e si dirige a destra, mentre digitando 180 la linea va dal centro verso il basso.

Va sottolineato il fatto che il computer accetta valori in gradi, che converte poi in radianti. Nei programmi (eccetto in quello degli Acorn) la variabile immessa è divisa per 180 e moltiplicata per pigreco (PI), evitando così tale conversione all'operatore.

```
10 BORDER 4: PAPER 4:INK 0: CLS
20 CIRCLE 131,88,60
30 PLOT 131,84: DRAW 0,8: PLOT 127, 88:
   DRAW 8,0
40 FOR a=0 TO 2*PI STEP PI/4
50 PLOT 131+55*SIN a,88+55*COS a
   : DRAW 10*SIN a,10*COS a
60 NEXT a
70 PRINT AT 2,16;0
80 FOR b = 45 TO 360 STEP 45
90 PRINT AT 10—10*COS (b/180*PI), 15+
   10*SIN (b/180*PI);b
100 NEXT b
110 INPUT "□□□□□
    ANGOLO (IN GRADI) DA□□□□□□□□□
    □□□VISUALIZZARE?",c
120 INK 2
130 PLOT 131,88: DRAW 45*SIN (c/180*PI),
    45*COS (c/180*PI)
140 INK 0
150 INPUT "□UN ALTRO ANGOLO (S/N)?";d$
160 IF d$="s" THEN PLOT 131,88 : DRAW
    OVER 1;45*SIN (c/180*PI), 45*COS
    (c/180*PI)
170 IF d$<>"s" THEN BORDER 7: PAPER
    7: CLS : STOP
180 PLOT 131, 84: DRAW 0,8: PLOT 127,88:
    DRAW 8,0: GOTO 110
```

Una versione di questo programma nel Basic standard del Commodore sarebbe molto complessa, mancando le più semplici istruzioni grafiche. Perciò questo programma (e tutti gli altri di questa lezione) è scritto utilizzando la cartuccia del Simon's BASIC.

```
5 INPUT"♡■ANGOLO";A:A=A/180*π
10 HIRES 1,0
20 CIRCLE 160,100,70,70,1
25 TEXT 160,20,"0",1,1,10
28 TEXT 157,97,"+",1,1,10
30 FOR X=45 TO 360 STEP 45
35 TEXT 140+90*SIN(X/180*π),100—90*
   COS(X/180*π),STR$(X),1,1,10
40 NEXT X
60 FOR X=0 TO 1.75*π STEP π/4
65 LINE 160+70*SIN(X),100—70*COS(X),
   160+60*SIN(X),100—60*COS(X),1
70 NEXT X
80 LINE 160,100,45*SIN(A)+160,100—45*
   COS(A),1
100 PAUSE 5:NRM:RUN
```

```
10 MODE1
20 MOVE 680,912
30 FOR A=0 TO 2*PI+.05 STEP .05
40 DRAW680—400*COS(A+PI/2),512+
   400*SIN(A+PI/2)
50 NEXT
60 VDU5
70 FOR A=PI/4 TO 2*PI STEP PI/4
80 MOVE 630—450*COS(A+PI/2),512+
   450*SIN(A+PI/2)
90 PRINT;INT(DEG(A)+.5)
100 MOVE 680—350*COS(A+PI/2),512+
    350*SIN(A+PI/2):DRAW 680—400*COS
    (A+PI/2),512+400*SIN(A+PI/2)
```

```
110 NEXT
120 VDU4
130 A2 = Ø
140 MOVE66Ø,512:DRAW7ØØ,512:MOVE68Ø,
    492:DRAW68Ø,532
150 VDU3Ø:PRINT″ "ANGOLO"STRING$(7,
    "□"):INPUT TAB(5,2);A
160 GCOLØ,Ø
170 MOVE 68Ø,512
180 DRAW68Ø − 35Ø*COS(A2 + PI/2),512 +
    35Ø*SIN(A2 + PI/2)
190 A = RAD(A)
200 MOVE68Ø,512
210 GCOLØ,3
220 DRAW68Ø − 35Ø*COS(A + PI/2),512 +
    35Ø*SIN(A + PI/2)
230 A2 = A
240 GOTO 140
```

```
10 PMODE4,1
20 PCLS
30 PI = 4*ATN(1)
40 CIRCLE(127,95),8Ø,5
50 FOR X = Ø TO 2*PI STEP PI/4
60 LINE(127 + 72*SIN(X),95 − 72*COS(X)) −
   (127 + 79*SIN(X),95 − 79*COS(X)),PSET
70 NEXT X
80 CLS:INPUT"ANGOLO DA CONVERTIRE□";Z
90 SCREEN 1,1
100 X = 127 + 6Ø*SIN(Z*PI/18Ø):Y = 95 − 6Ø*
    COS(Z*PI/18Ø)
110 LINE(127,91) − (127,99),PSET
120 LINE(123,95) − (131,95),PSET
130 LINE(127,95) − (X,Y),PSET
140 IFINKEY$ = "" THEN 14Ø
150 LINE(127,95) − (X,Y),PRESET
160 GOTO8Ø
```

Ogni programma, meno quello dello Spectrum, inizia predisponendo il modo grafico appropriato alla linea 1Ø e ripulendo lo schermo.

Si noti che nella versione Commodore è necessario immettere il valore dell'angolo *prima* che il computer disegni il goniometro perché, una volta in modo grafico, non si possono visualizzare scritte.

Dato che né il Dragon né il Tandy possiedono internamente pigreco, la linea 3Ø in questa versione assegna tale valore alla variabile PI.

A questo punto tutti i programmi disegnano il goniometro: su Commodore, Dragon, Tandy e Spectrum viene usato il comando CIRCLE (linea 2Ø per Commodore e Spectrum e la linea 4Ø per gli altri). Gli Acorn non hanno un comando CIRCLE, per cui il cerchio viene disegnato dalle linee 2Ø-5Ø, con una routine che sarà esposta più avanti.

Poi il programma passa ai segni che completano il cerchio, cioè collocano i numeri che indicano i gradi e le lineette per segnare esattamente la posizione di ogni angolo.

Questi segni sono visualizzati riferendosi alla loro posizione angolare, con i metodi descritti in seguito.

La parte seguente dei programmi (eccetto che per il Commodore), riguarda l'immissione dell'angolo scelto: la linea 13Ø per gli Acorn, la linea 8Ø per Dragon e Tandy e linea 11Ø per lo Spectrum. Dato che l'INPUT per il Commodore deve essere alla linea 5, all'inizio del programma, questa versione ridisegna il goniometro a ogni nuovo angolo.

Si noterà che, al centro dello schermo, c'è una crocetta che serve per controllare la misura degli angoli disegnati dal computer: se si richiede un angolo di 9Ø gradi, allora la linea orizzontale sulla destra della crocetta sarà coperta dalla nuova linea.

Su Spectrum, Acorn, Dragon e Tandy la crocetta viene disegnata con due linee, mentre il Commodore fa comparire un segno più centrato sullo schermo.

Visualizzato l'angolo desiderato, compare la richiesta per un altro angolo. Per segnare il nuovo angolo, sugli Acorn e sullo Spectrum viene prima cancellata la linea vecchia, mentre sul Dragon e sul Tandy essa viene cancellata dopo una pausa introdotta con il ciclo FOR...NEXT.

Nel Commodore ciò è inutile, poiché l'intero schermo è cancellato a ogni nuovo angolo.

Dragon, Tandy e Commodore fanno una pausa dopo aver disegnato l'angolo, prima di chiedere l'INPUT di un nuovo angolo. Gli Acorn, invece, passano subito a chiedere l'immissione di un nuovo angolo e, finché non la si effettua, l'ultima linea non viene cancellata. La versione Spectrum prevede una routine per chiedere se si vuol ricominciare, nel qual caso cancella l'ultima linea, altrimenti si ferma.

Per ottenere una bussola, si sostituiscono le indicazioni dei gradi con i punti cardinali N, S, E e O. Per abituarci all'orientamento, per esempio, si immette la direzione in gradi e sullo schermo compare la linea che indica la direzione da prendere. Se si vuole provare, si individuino le linee di programma che si occupano della visualizzazione di ciò che circonda il cerchio.

PUNTI SUL CERCHIO

Tutti i programmi visti usano SIN e COS, due funzioni BASIC che stanno per 'seno' e 'coseno'. Queste sono due delle funzioni trigonometriche del computer: non ci si lasci ingannare dal nome altisonante, perché il concetto su cui si basano è veramente semplice.

Esse si riferiscono alla posizione di un punto sulla circonferenza di un cerchio, in relazione a due assi che, in sostanza, indicano quanto a destra o a sinistra e quanto in alto o in basso si trova il punto.

Le due linee che formano la crocetta al centro del cerchio sono gli assi, la verticale è detta asse 'y' e l'orizzontale asse 'x'.

Osservando il punto **A** sul diagramma (**figura 2**) si vede che una linea lo collega a ogni asse: queste linee sono indicate con **p** e **q**. Si nota subito che, nell'esempio, **p** e **q** sono della stessa lunghezza. Abbassando **A** verso destra sulla circonferenza, la lunghezza di **p** aumenta, mentre quella di **q** diminuisce.

Quando **A** raggiunge il punto segnato con 9Ø, **q** è Ø, mentre **p** è uguale al raggio del cerchio.

Possiamo anche usare il programma del goniometro, per renderci meglio conto del fatto. Si chiedano al computer gli angoli Ø, 3Ø, 45 e 9Ø. Ad ogni nuova linea tracciata, si riesce facilmente a immaginare come **p** e **q**, benché non compaiano, cambino in lunghezza.

IL VALORE DI SIN E COS

A ogni angolo dato, corrisponde sempre lo stesso rapporto tra **p** e **q** e il raggio del cerchio: quando, ad esempio, l'angolo è di 9Ø, **p** è uguale al raggio. Come nel caso precedente, questo rapporto rimane identico per un dato angolo e cambia al variare dell'angolo.

Il rapporto tra il raggio e **p** è detto 'seno' dell'angolo. Il rapporto tra il raggio e **q** è detto 'coseno' dell'angolo. Per ottenere il seno o il coseno si divide la lunghezza della linea (**p** o **q**) per il raggio. Se il raggio è di lunghezza unitaria, allora il valore del seno e del coseno è *uguale* alla lunghezza di **p** e **q**.

Il triangolo di **figura 3** è ripreso dal diagramma del cerchio. La figura a cuneo è formata dall'angolo tra l'asse x e la linea che congiunge il punto **A** al centro del cerchio. Una terza linea, perpendicolare, va da **A** all'asse x, formando il terzo lato del triangolo.

Su questo triangolo il seno è il rapporto tra il lato *opposto* all'angolo al centro e l'*ipotenusa*. L'ipotenusa è sempre il lato opposto all'angolo retto (vedi **figura 3**). Il terzo lato è detto lato adiacente. Analogamente, il coseno è il rapporto tra l'altra coppia di lati. Rimane un'ultima combinazione di lati, il cui rapporto è detto la tangente dell'angolo. I rapporti sono quindi:

seno = lato opposto/ipotenusa
coseno = lato adiacente/ipotenusa
tangente = lato opposto/lato adiacente

Tutti e tre i rapporti si possono calcolare con il computer, mediante le funzioni SIN, COS e TAN. Per esempio, mediante PRINT SIN.5 si ottiene sullo schermo il seno di un angolo di Ø,5 radianti.

Tornando al diagramma del cerchio, se si sposta il punto in senso orario, i valori del seno e del coseno diventano, rispettivamente, più grande più piccolo, col mutare di dimensione di **p** e **q**. Dopo i 9Ø gradi, però, il seno inizia a decrescere e quando il punto passa i 18Ø gradi, l'andamento si ribalta nuovamente.

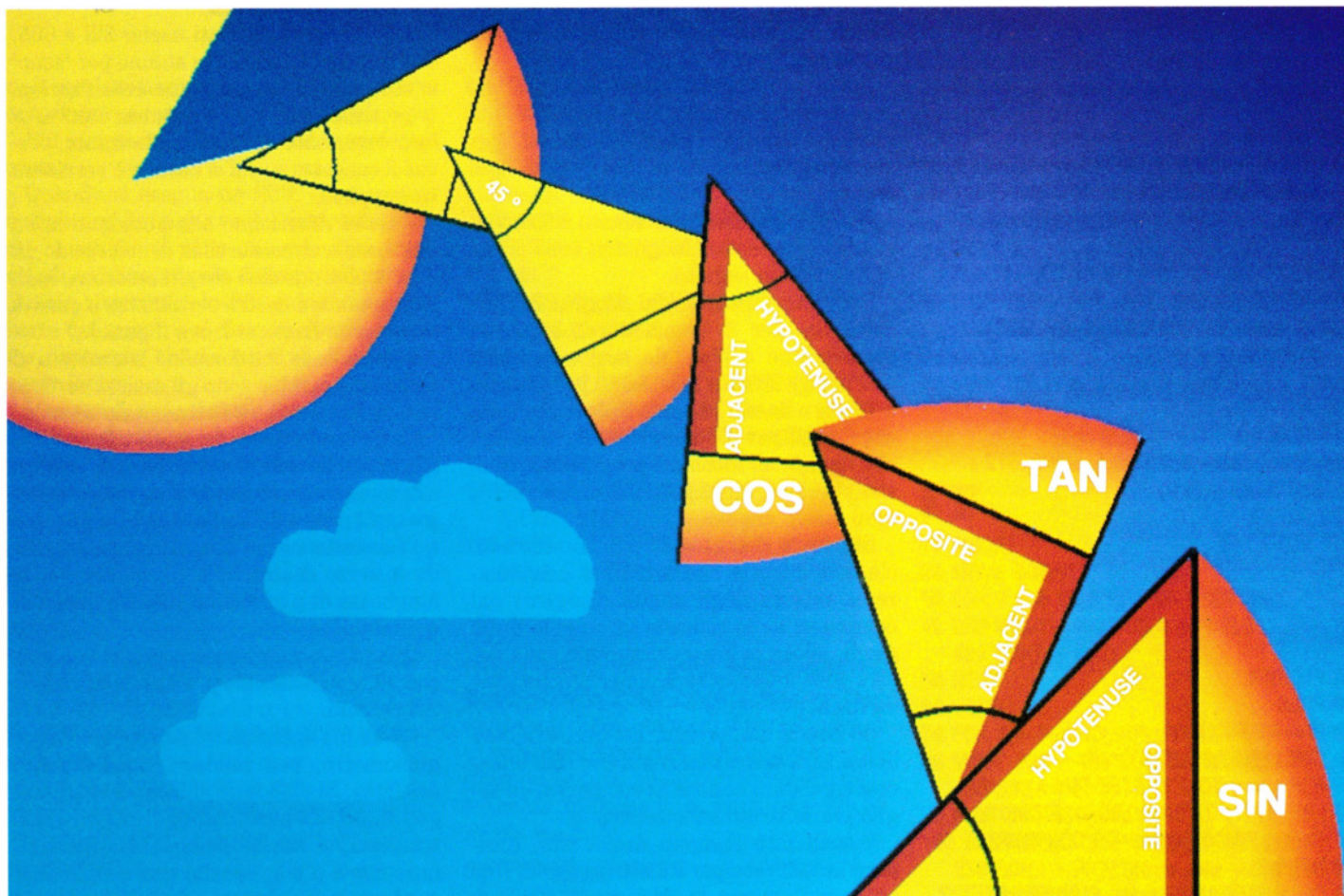
Ricordiamo che il seno misura quanto a *destra*, rispetto all'asse 'y', si trovi un punto. Ciò significa che, nella *metà* a *sinistra* del cerchio dove il punto si trova a sinistra dell'asse 'y', il seno è *negativo*.

Similmente, se il punto va nella parte inferiore del cerchio, cioè sotto l'asse 'x', il coseno diviene negativo, poiché il coseno misura quanto *sopra*, rispetto all'asse 'x', si trova il punto.

ANDAMENTO GRAFICO DI SIN E COS

L'andamento delle funzioni seno e coseno, al variare dell'angolo al centro, risulta più chiaro disegnando un grafico dei loro valori per una serie di angoli. Il seguente programma ha questo compito.

```
10 PLOT Ø,88
20 DRAW 255,Ø
30 PLOT 20,Ø: DRAW Ø,175
```

3. Il rapporto tra le lunghezze dei lati di ogni triangolo rettangolo è fissato dagli altri angoli. A seconda dei lati confrontati, il rapporto prende nome di seno, coseno o tangente

4. Disegno di un goniometro sul Commodore

5. Il goniometro dello Spectrum mostra un angolo di 7Ø°

```
40 PLOT 1Ø,158: DRAW 15,Ø: PLOT 1Ø,18:
   DRAW 15,Ø
50 PRINT AT 2,Ø;1;AT 2Ø,Ø;—1;AT 11,Ø;Ø;
   AT 2Ø,15;18Ø;AT 2Ø,29;36Ø
60 FOR a=Ø TO 2*PI STEP .Ø6
70 PLOT 2Ø+a*35,88+7Ø*SIN a
80 PLOT INK 2;2Ø+a*35,88+7Ø*COS a
90 NEXT a
```

```
10 HIRES Ø,1:MULTI 3,7,1:COLOUR 6,6:
   LINE 15,1ØØ,36Ø,1ØØ,3
15 LINE2Ø,Ø,2Ø,2ØØ,3:TEXT 1,5,"+1",1,3,6
18 TEXT 1,17Ø,"—1",2,3,6:TEXT 6,9Ø,"Ø",3,3,
   6
20 FOR Z=Ø TO 2*π STEP.Ø5
30 PLOT Z*22+2Ø,1ØØ—SIN(Z)*8Ø,1
35 PLOT Z*22+2Ø,1ØØ—COS(Z)*8Ø,2
40 NEXT Z
50 GOTO 5Ø
```

```
10 MODE1
20 DRAW Ø,1Ø24
30 MOVEØ,512:DRAW 128Ø,512
40 VDU5:MOVEØ,744:PRINT"1"
50 MOVEØ,31Ø:PRINT"—1"
60 MOVE Ø,5ØØ
70 FOR T=Ø TO 36Ø STEP 9Ø
80 PRINT;T;
90 PLOT Ø,18Ø,Ø
100 NEXT
110 VDU4
120 X=Ø
130 MOVE Ø,512
140 FOR T=Ø TO 2*PI+.1 STEP .1
150 DRAW X,512+2ØØ*SIN(T)
160 X=X+17
170 NEXT
180 MOVE Ø,712
190 X=Ø
200 GCOLØ,1
210 FOR T=Ø TO 2*PI+.1 STEP .1
220 DRAW X,512+2ØØ*COS(T)
230 X=X+17
240 NEXT
```

```
10 PMODE3,1
20 PCLS
30 PI=4*ATN(1)
50 LINE(6,95)—(255,95),PSET
60 LINE(1Ø,Ø)—(1Ø,191),PSET
70 LINE(6,45)—(1Ø,45),PSET
80 LINE(6,145)—(1Ø,145),PSET
90 SCREEN1,1
100 FOR X=72 TO 255 STEP 61
110 LINE(X,92)—(X,95),PSET
120 NEXT
130 FOR X=Ø TO 2*PI STEP PI/123
140 PSET(123*X/PI+1Ø,95—5Ø*SIN(X),3)
150 PSET(123*X/PI+1Ø,95—5Ø*COS(X),2)
160 NEXT
170 GOTO 17Ø
```

Come per il programma del cerchio, le versioni per Acorn, Commodore, Dragon e Tandy iniziano predisponendo il modo grafico appropriato alla linea 1Ø. Sul Dragon e sul Tandy, alla linea 3Ø viene definita la variabile pigreco.

Tutti i computer disegnano quindi gli assi per il grafico e tutti, meno il Dragon e il Tandy, corredano il disegno di scritte: sull'asse y compaiono i valori 1, Ø e −1 (l'intera gamma possibile), mentre sull'asse x appaiono i gradi, da Ø a 36Ø.

Dragon e Tandy non possono visualizzare scritte sullo schermo grafico ad alta risoluzione, per cui l'identificazione avviene graduando gli assi con lineette e non con numeri.

Poi il computer passa a disegnare il grafico, uno per il seno e uno per il coseno. Un ciclo FOR...NEXT provvede a fornire i vari valori angolari, prima di calcolare SIN e COS. Come si è visto, ci sono 2*PI radianti in un cerchio. Quindi, per ottenere ogni possibile angolo, serve un ciclo FOR T=Ø TO 2*PI.

Tuttavia, per non allungare troppo l'esecuzione, è stato introdotto uno STEP in modo da non far disegnare al computer proprio tutti gli angoli possibili, per un giro completo (2 pigreco radianti o 36Ø gradi) del cerchio.

Con STEP il programma si sveltisce molto su tutti i computer, meno che sugli Acorn, anche se la linea risultante è punteggiata. Si provi a togliere STEP per apprezzare la differenza.

Le linee cruciali, per il calcolo di SIN e COS (le 15Ø, 22Ø sugli Acorn; le 3Ø, 35 sul Commodore; le 14Ø, 15Ø sul Dragon e sul Tandy e le 7Ø, 8Ø sullo Spectrum) sembrano complicate, ma sono in realtà semplici. Ogni linea disegna il grafico attraverso i comandi PLOT, PSET o DRAW: gli Acorn disegnano una linea piena, mentre gli altri computer una serie di punti.

Le posizioni di DRAW o di PLOT sono determinate dal SIN o COS che compare nell'espressione. La serie di altri numeri, apparentemente incomprensibili, serve semplicemente per portare il risultato di SIN o COS in una scala adatta alle dimensioni dello schermo. Siccome ciascun computer ha una presentazione video diversa e diversi comandi grafici, anche questi calcoli differiscono.

DISEGNARE CERCHI

Si è visto come calcolare la posizone di un punto sulla circonferenza di un cerchio in

riferimento agli assi x e y, conoscendo l'angolo al centro e *anche* il raggio.

Ma si può seguire un percorso diverso, per individuare la posizone del punto, ricavandola dagli altri elementi e in ciò stà la chiave per disegnare cerchi sullo schermo o per disporvi attorno caratteri o spazi come sul programma per la misurazione degli angoli.

Occorre un programma che fornisca al computer un centro e un raggio per il cerchio e che calcoli le coordinate x e y per ogni angolo dato. Cosicché, immettendo una serie di angoli, il computer saprà individuare la corrispondente serie di punti sulla circonferenza scelta. Il trucco stà nel far calcolare al computer le coordinate x e y mediante SIN e COS.

Si digiti questo breve programma:

```
20 FOR x=Ø TO 2*PI STEP PI/3Ø
30 PLOT 128+5Ø*SIN x,88+5Ø*COS x
40 NEXT x
```

Lanciando il programma, si vede una serie di punti che si dispongono rapidamente a formare un cerchio. Il tracciato potrebbe anche essere continuo, ma ciò richiederebbe molto tempo.

In ogni programma, meno quello dello Spectrum, il computer prima di tutto seleziona il modo grafico. Su Dragon e Tandy, va definito anche il valore di pigreco (PI). Poi, un ciclo FOR...NEXT fa calcolare al computer una serie di angoli da Ø a 2*PI (linea 3Ø su Spectrum, Dragon e Tandy, 2Ø su Commodore, 4Ø su Acorn), cioè un cerchio completo. Il programma è sveltito dal particolare valore di STEP, generando però una linea sensibilmente discontinua.

La linea successiva fa disegnare al computer un punto per ogni angolo, determinandone la posizione con la formula:
PLOT raggio * SIN X, raggio * COS X

Si tenga presente il diagramma del cerchio alla **figura 2** e come queste funzioni determinino le coordinate per ogni angolo dato. Gli altri numeri aggiunti o sottratti in questa linea di programma fanno sembrare la formula più complicata, ma non si tratta che di valori necessari a individuare il centro del cerchio, in relazione allo schermo grafico del computer. Sugli Acorn, il centro si trova a 68Ø,512; sul Commodore a 15Ø,1ØØ; su Dragon e Tandy a 127,95 e sullo Spectrum a 128,88.

Per acquisire familiarità con queste funzioni, si provi a cambiare la posizione del centro (sugli Acorn si modifichi anche la linea 3Ø, che rappresenta il centro più il raggio). Oppure si cambi la grandezza del cerchio, modificando il raggio: linea 2ØØ sugli Acorn, 8Ø sul Commodore, Dragon e Tandy e 5Ø sullo Spectrum.

L'aspetto del cerchio può essere cambiato intervenendo anche su altri valori: alla prima linea del ciclo, lo STEP determina l'intervallo tra ogni punto: più lo si riduce, più la linea del cerchio è compatta. Per un numero maggiore di punti, si diminuisca il valore di STEP o si divida PI per un numero più grande.

```
10 HIRES Ø,1
20 FOR X=Ø TO 2*π STEP.Ø1
30 PLOT SIN(X)*8Ø+15Ø,COS(X)*8Ø+1ØØ,1
40 NEXT X
50 PAUSE 1Ø
```

```
20 MODE1
30 MOVE 68Ø,712
40 FOR X=Ø TO 2*PI+.Ø5 STEP .Ø5
50 PLOT 69,68Ø+2ØØ*COS(X+PI/2),512+
   2ØØ*SIN(X+PI/2)
60 NEXT
```

```
10 PMODE 4,1:PCLS:SCREEN1,1
20 PI=4*ATN(1)
30 FOR X=Ø TO 2*PI STEP PI/45
40 PSET(127+8Ø*SIN(X),95—8Ø*COS(X),5)
50 NEXT X
60 GOTO 6Ø
```

DISEGNARE UN'ELLISSE

C'è un'altra modifica interessante da fare: cambiare il cerchio in un'ellisse. Per far ciò si renda il numero per cui si moltiplica SIN più grande o più piccolo del numero per cui si moltiplica COS.

Moltiplicando il seno per un numero maggiore si ottiene un'ellisse larga e bassa; viceversa, moltiplicando il coseno per un numero maggiore si ottiene un'ellisse più alta e più stretta.

Il prossimo programma disegna una serie di ellissi che producono l'effetto di una sfera.

```
10 FOR z=Ø TO 5Ø STEP 5
20 FOR x=Ø TO 2*PI STEP PI/15
30 PLOT 128+z*SIN x,88+5Ø*COS x
40 NEXT x
50 NEXT z
```

```
10 HIRES Ø,1
15 FOR Z=Ø TO 1ØØ STEP 1Ø
20 FOR X=Ø TO 2*π STEP.Ø5
30 PLOT SIN(X)*Z+15Ø,COS(X)*6Ø+1ØØ,1
40 NEXT X,Z
50 PAUSE 1Ø
```

```
10 MODE 1
20 MOVE 68Ø,712
30 FORZ=Ø TO 2ØØ STEP 4Ø
40 FORX=Ø TO 2*PI+.Ø5 STEP .Ø5
50 PLOT 69, 68Ø+Z*COS(X+PI/2),512+
   2ØØ*SIN(X+PI/2)
60 NEXT:NEXT
```

```
10 PMODE4,1:PCLS:SCREEN1,1
20 PI=4*ATN(1)
30 FOR Z=Ø TO 8Ø STEP5
40 FOR X=Ø TO 2*PI STEP PI/45
50 PSET(127+Z*SIN(X),95—8Ø*COS(X),5)
60 NEXT X
70 NEXT Z
80 GOTO8Ø
```

I programmi per la sfera sono un lieve adattamento del programma per il cerchio.

Invece di disegnare un'ellisse, se ne disegna un'intera serie con un secondo ciclo FOR...NEXT, durante il quale viene variata la grandezza del numero per cui si moltiplica la coordinata x. Il computer disegna così una sequenza di ellissi di diversa grandezza.

L'ellisse può essere fatta crescere a piacimento, intervenendo sulla grandezza e sullo STEP di Z, la variabile di controllo.

Quali le dimensioni massime di un cerchio disegnato dal computer?
Il raggio non deve superare metà della dimensione minore dello schermo. Nei vari computer il raggio massimo è: Acorn 512, Commodore 1ØØ, Vic 88, Dragon e Tandy 95, Spectrum 88, purché il centro del cerchio sia in posizione centrata sullo schermo. Se il cerchio è troppo largo nello Spectrum si produce un errore del tipo 'Integer out of range' (numero intero fuori gamma). Gli altri computer disegnano quanto del cerchio entra sullo schermo. Il comportamento del Commodore 64, in un simile evento, varia da caso a caso.

ERRATA CORRIGE

Pag. 8, colonna 3, 4ª riga dall'alto
PRINT AT 1Ø,1Ø; <ABC grafiche>

Pag. 9, colonna 1, 10ª riga dal basso
Le linee da 1ØØ a 16Ø ...

Pag. 11, colonna 2, 2ª riga
... CHR$ (226)

Pag. 28, colonna 3, 5ª riga dall'alto
sostituire MODO 7 con MODE 7

Pag. 35, box D+R, riga 7Ø
7Ø IF X=Y THEN PRINT "FUORI TEMPO": IF conto >Ø THEN LET conto=conto −1

Pag. 85, colonna 2, 4ª riga
DRAW Ø, −5Ø

Pag. 87, colonna 1, fondo colonna
1Ø HIRES □ 1,Ø (da aggiungere)

Pag. 4, colonna 2, linee 6Ø, 8Ø
6Ø IF G=X THEN PRINT "RISPOSTA ESATTA"
8Ø IF G< >X THEN PRINT "ERRATO-RIPROVARE"

Pag. 5, colonna 1, linea 9Ø
digitare RETURN al posto di ENTER

Pag. 7, colonna 1, linea 8Ø
8Ø PRINT "QUANTO FA □";N;"□ PER 9?"

Pag. 15, colonna 2, linee 11Ø, 1ØØØ, 1Ø5Ø
nella linea 11Ø digitare 7.5 al posto di 7,5 - nella linea 1ØØØ DATA Ø,128,Ø,1,192,Ø,1,176,Ø - nella linea 1Ø5Ø digitare 14 al posto di 14Ø

Pag. 31, colonna 2, riga 6Ø
6Ø FOR D=1TO5Ø:NEXT

Pag. 48, 49, 5Ø
In 6ØØØ digitare LETTURA al posto di SCRITTURA
2 GOSUB 6:GOTO 1ØØ
eliminare linea 42
78 DATA "□CREAZIONE FILE", "IMMISSIONE RECORD □", "VISIONE RECORD □□□"
8Ø DATA "□RICERCA RECORD □", "SCRITTURA FILE □", "LETTURA FILE"
5ØØ PRINTX1$"1"X2$M$(1)X3$;
111Ø V=INT(FR/(LN+5+3*NF))
112Ø PRINT V "□ RECORD DISPONIBILI":FOR D=1TO15ØØ:NEXT
138Ø R(U2)=RU:IFA>2THEN UP=U2:PRINT CHCCRV$"□ RECORD N.□" UP+1:GOTO 31ØØ
3Ø2Ø FOR UP=UØTOU−1
1311Ø POKEP1, PEEK (P1) AND127:P1=P1+1:PRINTA$;:IX$=IX$+A$:GOTO 13Ø2Ø

Pag. 58, colonna 2
6Ø IF K$="S" THEN P=P−1 :GOTO 9Ø
in linea 7Ø digitare D al posto di R

Pag. 59, colonna 1, riga 15Ø
digitare ; al posto di :

Pag. 79, colonna 3
4Ø6Ø z=Ix(ix):GOSUB 13ØØØ:IF RIGHT$(ix$,1)="□" THEN ix$=LEFT$(ix$,LEN(ix$)−1)

Pag. 83, righe 2ØØ, 22Ø
aggiungere , Ø in fondo alle due righe

Pag. 87, colonna 2, riga 1Ø
aggiungere la , tra 1 e Ø

Pag. 93
2Ø LET LRE=Ø
15 IFK$="N" THEN LET LRE= LRE+5Ø:LET LITRI=LITRI+QTA
16Ø PRINT"■■■■■■■■"; TAB (14) LITRI
17Ø PRINT"■■■■■■■■■■■■" TAB (14) LRE

Pag. 96, colonna 1
sostituire riga 2 con
2Ø LET A$="□□□□□ BUONA PASQUA A TUTTI □□□□□□□□□□□□□□□□"
in linea 55 sostituire π con ■

Pag. 1ØØ, colonna 1, riga 27Ø
aggiungere IF prima di M=

Pag. 1Ø6, colonna 3
aggiungere la riga 9Ø GETK$:IFK$=""THEN GOTO 9Ø

Pag. 128, riga 4ØØ
aggiungere la (dopo LEN.

Pag. 159, colonna 3, riga 18Ø, 44Ø, 46Ø
sostituire (V$,() con (V$,8)
sostituire VAH con VAL
sostituire VALI$) con VAL (I$)

Pag. 164, colonna 1, riga 3
digitare 24Ø al posto di 14Ø

Pag. 171, colonna 2, riga 15
digitare POKE Z,Ø al posto di POKE,Ø

Pag. 171, colonna 2, primo listato
il numero di riga dopo la 45 deve essere 5Ø e non 3Ø

Pag. 196, colonna 1
in riga 1Ø16 sostituire 1Ø15 con 1Ø16
in riga 2ØØ6 inserire ■ tra le virgolette

Pag. 142, 143
in riga 11Ø sostituire 11ØØ con 11Ø
in riga 16Ø sostituire GOTO con GOSUB
in riga 17Ø inserire i : dopo D$(4,1)
in riga 18Ø inserire (S/N) dopo SEI SICURO
in riga 24Ø inserire ◫ nei due spazi vuoti
in riga 33Ø aggiungere la) alla fine della riga
in riga 35Ø aggiungere la) alla fine della riga
in riga 37Ø aggiungere ◫ tra le virgolette finali ed il ; alla fine riga
in riga 38Ø aggiungere ◫ tra le virgolette finali
in riga 4ØØ eliminare ".ØØ"
in riga 49Ø sostituire QQS con QQ$
in riga 54Ø eliminare IMMETTERE e aggiungere il ; tra " e D$(1,C1)
in riga 55Ø aggiungere ■ dopo ■
in riga 56Ø aggiungere □□ prima di IMPORTO
in riga 85Ø dopo PRINT aggiungere le "
in riga 94Ø eliminare QUALSIASI e OPPURE
57Ø IFQQ$=CHR$(13) THEN 6Ø

Pag. 3, colonna 3, riga 6Ø
digitare THEN al posto di THE

Pag. 9, colonna 1
2Ø CLEAR 65199:LET B=652ØØ:LET Z=1

Pag. 32, colonna 2, riga 1Ø
aggiungere le " al termine della riga

Pag. 44, colonna 1, riga 4Ø
digitare n al posto di N

Pag. 44, colonna 3, riga 3Ø
3Ø DATA BIN Ø, BIN Ø, BIN Ø, BIN ØØØØØ111, BIN ØØØØØØ11, BIN 11111111, BIN Ø1111111, BIN ØØ111111

Pag. 5Ø, 51
5ØØ LET I$=INKEY$:IFI$=""THEN GOTO 5ØØ
7ØØØ PRINT AT 1Ø,8; "SEI SICURO (S/N)?":IF INKEY$=""THEN GOTO 7ØØØ
in 951Ø sostituire AT 2Ø,U con AT 18,U

Pag. 57, colonna 3, riga 1ØØ
digitare INKEY$ al posto di INEKY$

Pag. 63, colonna 3, riga 16Ø
digitare RDN al posto di RND

Pag. 78, colonna 1, riga 9Ø3Ø
eliminare un ;

Pag. 81, colonna 3, riga 1Ø3Ø
sostituire 1ØØ con 1Ø7

Pag. 1Ø3, colonna 2, riga 14Ø
sostituire > con <

Pag. 112, colonna 2, riga 5Ø
aggiungere il ; prima di LINE b$

Pag. 13Ø, colonna 3, riga 3Ø
sostituire ns(n) con n$(n)

Pag. 139, colonna 1, riga 1Ø1Ø
sostituire IMK7 con INK7

Pagina 139, riga 22Ø
sostituire QUALE SCEGLI con SELEZIONE

Pag. 15Ø, colonna 1, riga 62Ø
sostituire 9 con .9

Pag. 95, colonna 1
2Ø LET B$="□VUOI"
in linea 4Ø digitare ; al posto di :
in linea 5Ø digitare ; al posto di :
N.B. In tutti i programmi validi anche per lo ZX SPECTRUM, le istruzioni poste sulla stessa linea, vanno spezzate in più righe.

Pag. 13, colonna 1, riga 2Ø
2Ø FOR I=32ØØØ TO 3211Ø

Pag. 13, colonna 2, riga 13Ø
13Ø DATA 255,255,255,235,65,1Ø2, Ø,Ø,255,255,255, 174, 6, 1ØØ,Ø,Ø

Pag. 14, colonna 1, riga 6Ø
eliminare il ,Ø finale

Pag. 19, colonna 2, riga 4Ø
digitare le , al posto dei

Pag. 21, colonna 1, riga 5Ø
5Ø LINE(255,191) − (255 − L,Ø), PSET

Pag. 27, colonna 1, riga 9Ø
9Ø PRINT @ 173, CHR$(141):PRINT @ 177, CHR$(141)

Pag. 51, colonna 3, riga 1Ø7Ø
digitare N$(A) al posto di NS(A)

Pag. 57, colonna 2, righe 6Ø, 7Ø
digitare S al posto di L
digitare D al posto di R

Pag. 59, colonna 3, riga 5Ø
digitare BL$ al posto di BLS

Pag. 82, colonna 3
41Ø IFDT=1 GOTO 47Ø

Pag. 94, colonna 1
17Ø PRINT @ 272, LIRE

Pag. 96, colonna 1, riga 2Ø
digitare = al posto del .

Pag. 129, colonna 3
5 DIM NOME$(5),ETÀ(5)

Pag. 14Ø, colonna 3, riga 114Ø
sostituire LENGHT con THEN

Pag. 4, colonna 3, linea 9Ø
digitare RETURN al posto di ENTER

Pag. 11, colonna 3, linea 27ØØ
il numero di riga è 27Ø e non 27ØØ

Pag. 12, colonna 1, linea 31Ø
eliminare il ,Ø finale

Pag. 19, colonna 2, linea 4Ø
aggiungere al termine della linea il ;

Pag. 28, colonna 3, riga 2Ø
digitare : al posto di ;
riga 5Ø
eliminare la , dopo la Y

Pag. 29, colonna 3, riga 4Ø (secondo programma)
4Ø PRINT TAB (X,1Ø); "□OOO<"

Pag. 52, colonna 3, riga 145
145 IF G< >7 THEN 148

Pag. 53, colonna 3, riga 9Ø24
il numero della riga deve essere modificato in 9Ø42

Pag. 62, colonna 1, riga 14Ø
inserire : dopo A$

Pag. 94, colonna 1
16Ø PRINT TAB(23,9);litri
17Ø PRINT TAB(23,1Ø);lire
2ØØ GOTO 13Ø

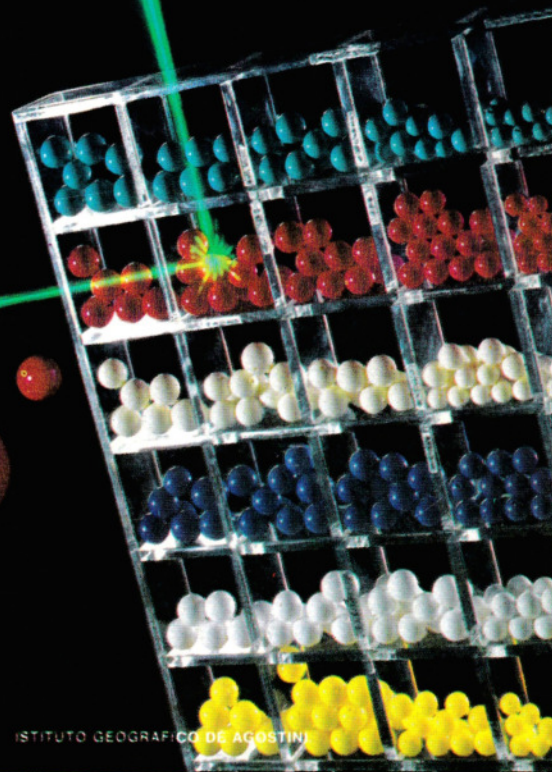
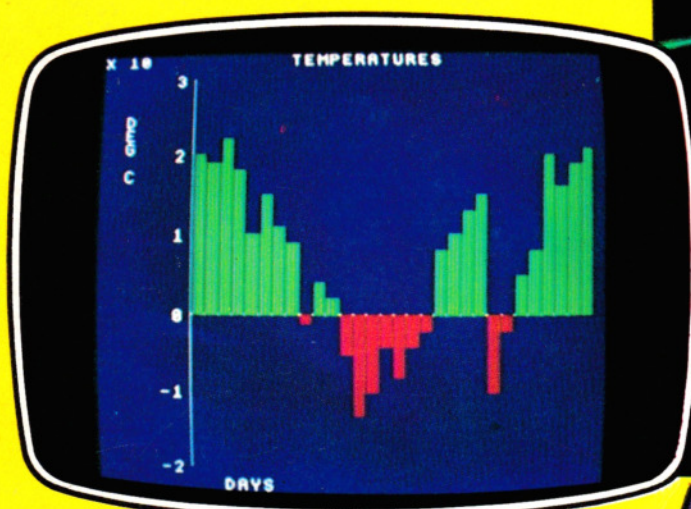
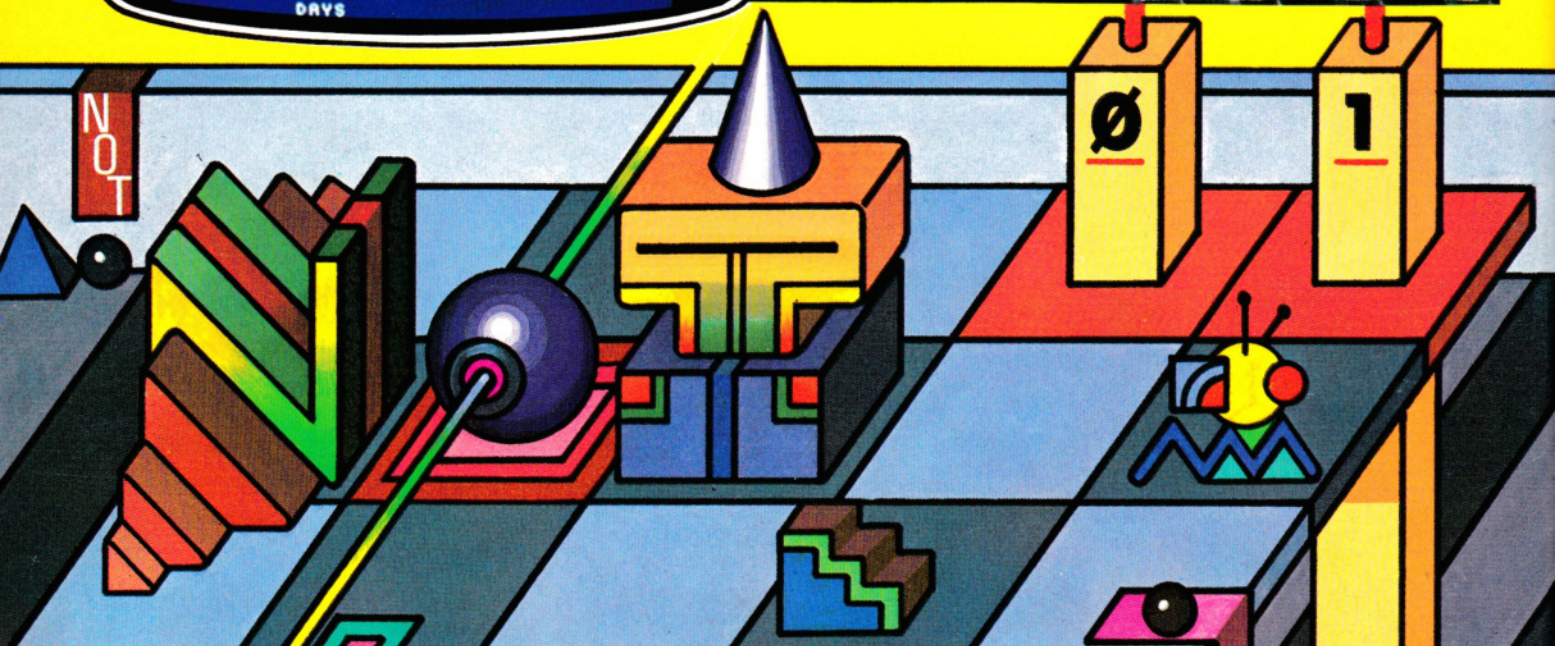
Pag. 96, colonna 1, linea 6Ø
digitare 2 al posto di 12

Pag. 1Ø1, colonna 3, riga 34Ø
sostituire "N□"; con "IN□";

Pag. 138
5ØØ PRINT'"TOTALE DELLE SPESE□:"; SPESE
56Ø DEF PROCVIS
92Ø DEF PROCSTAMPA
97Ø DEF PROCMODIF

Pag. 197, colonna 3, riga 39Ø
eliminare la) dopo 38

Pag. 2Ø1, colonna 2
eliminare linea 15Ø

Pag. 182, colonna 1, righe 11Ø, 12Ø
digitare 1Ø al posto di 1Ø4

NEL PROSSIMO NUMERO
□ Visualizzare fatti e cifre in modo professionale con i GRAFICI A BARRE.
□ Scopriamo i segreti dei GIOCHI D'AVVENTURA in una nuova serie di lezioni.
□ Gli OPERATORI LOGICI possono ampliare le capacità decisionali del computer.
□ Un programma MONITOR per preparare ed eseguire programmi in codice macchina.
□ La gestione di grandi quantità di dati è facilitata dall'impiego di MATRICI A PIÙ DIMENSIONI.
INPUT
9
CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER
ISTITUTO GEOGRAFICO DE AGOSTINI
X 10
TEMPERATURES
DEG C
3
2
1
0
-1
-2
DAYS
NOT
Ø
1
CHIEDETE INPUT AL VOSTRO EDICOLANTE